Stochastic Control

Neil Walton

May 14, 2020

Contents

Chapter 1

Optimal Control

- Dynamic Programs; Markov Decision Processes; Bellman's Equation; Complexity aspects.
- Discrete Time Merton Portfolio Optimization.
- Infinite Time Horizon Control: Positive, Discounted and Negative Programming.
- Algorithms: Policy Improvement & Policy evaluation; Value Iteration; Policy Iteration; Temporal Differences and Q-Factors.
- Optimal Stopping; One-Step-Look-Ahead.

1.1 Dynamic Programming

- Definition of Dynamic Program.
- Bellman's Equation.

The Basic Idea.

Let's discuss the basic form of the problems that we want to solve. See Figure 1.1. Here there is a controller (in this case for a com-

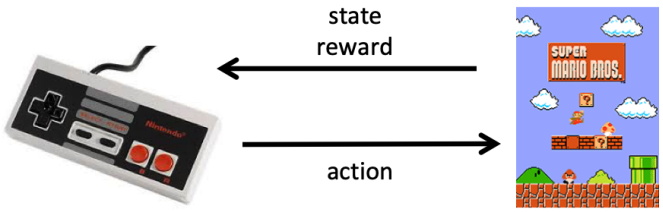


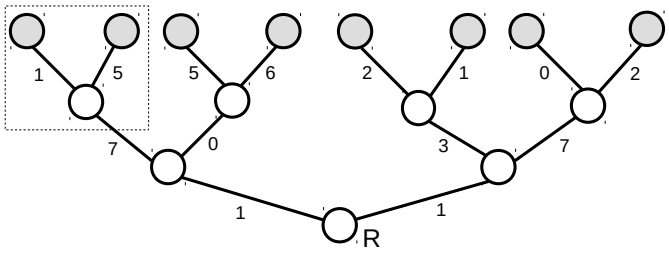
Figure 1.1: A control loop.

puter 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.

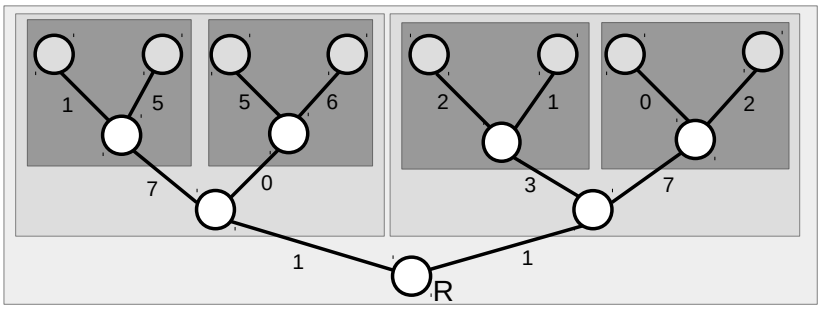
An Introductory Example

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.

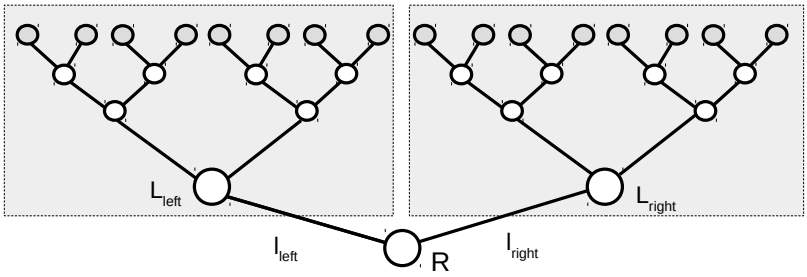


From this we see the optimal path has a cost of 5 and consists of going right, then left, then right.

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

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

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 $t = 0, 1, ..., T$. 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

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

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

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

$$x_{t+1} = f(x_t, \pi_t). \qquad (1.1)$$

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

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

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

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

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

[1] Since 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*

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

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 $t = T-1, ..., 0$*

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

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

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

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

□

Some Code

Below is some code (in Python) to solve a dynamic program. Notice we can solve the dynamic program by repeatly 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 [3]

A good example is to see what this means is to consider shortest paths and longest paths in an undirected graph. See Figure 1.2. 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 is 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.

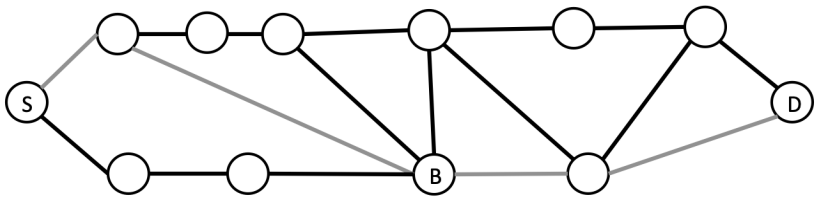


Figure 1.2: Shortest path from S to D.

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 thee 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 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

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

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

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

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 $t = 1, ..., T$. 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

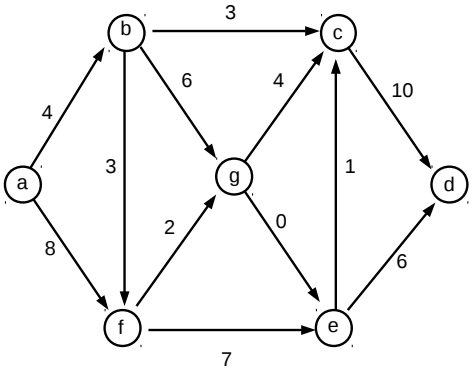
$$L_i(t + 1) = \min_{j:(i,j) \in E} \left\{ c_{ij} + L_j(t) \right\}, \qquad \text{for } i \neq d.$$

(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

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

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 $i = 1, ..., N$ requires t_i units of time and when completed has reward r_i. Given discount factor $\beta \in (0, 1)$, the total reward arragning the appointments in order $1, 2, ..., N$ is*

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

i) Write down a dynamic program for the optimal discounted reward. (Here let $W(S)$ be the optimal reward when $S \subset \{1, ..., N\}$ 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*

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

Here R and Q are positive semi-definate matrices.

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

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

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

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

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 Riccarti Recursion (the discrete time analogue of the Riccarti 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 $par(j) = \{i : (i, j) \in E\}$ have been completed. We also let $ch(j) = \{k : (j, k) \in E\}$ be the children of task j.*

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

i) Show that

$$\begin{aligned} EST(j) &= \max_{i \in par(j)} \{c(i) + EST(i)\}, \\ EFT(j) &= c(j) + \max_{i \in par(j)} EFT(i), \end{aligned}$$

with $EST(j) = 0$ and $EFT(j) = c(j)$ if $par(j)$ 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

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

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

Task	*Time*	*Parents*
a	*1*	-
b	*3*	*a*
c	*1*	*a*
d	*2*	*c*
e	*1*	*c*
f	*3*	*b,c*
g	*2*	*d,e*
h	*1*	*f, g*

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 equaion 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 [3]. The textbooks of Whittle [46] and Bertsekas [5] are noteworthy modern treatments of the field.

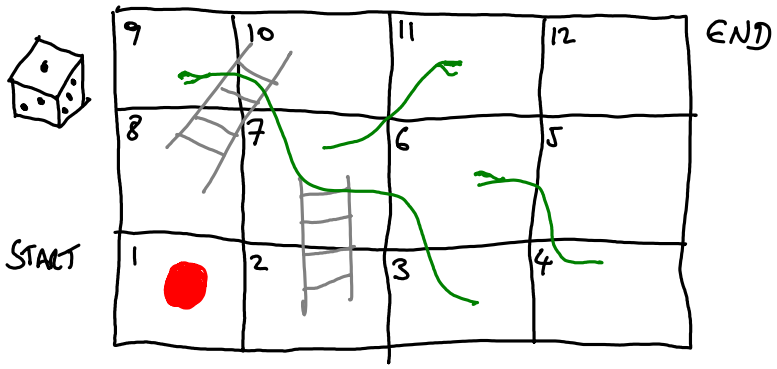
1.2 Markov Chains: A Quick Review

- Discrete-time Markov chains; The Markov property.
- Jump-chain construction; Potential Theory; Martingale Problems.

Introductory example: snakes and ladders

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. We give an initial example to better position our intuition.

Below in Figure 1.2, we are given a game of snakes and ladders (or shoots and ladders in the US). Here a counter (coloured 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 snakes 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 each movement of the counter from one state 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

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 postive 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 Vand 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 $x_0, ..., x_{t+1} \in \mathcal{X}$,*

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

and

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

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 $(X_1, ..., X_{t-1})$ and future X_{t+1} are conditionally independent of the present X_t. Otherwise stated, is says that, when we know the past and present states $(X_1, ..., X_t) =$

$(x_0, ..., x_t)$, the distribution of the future states $X_{t+1}, X_{t+2}, ...$ 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.

The following proposition shows that the evolution of a Markov chain can be constructed from its the current state and an independent "dice throw".

Prop 12 (Constructing Markov Chains)**.** *Take a function $f : \mathcal{X} \times [0,1] \to \mathcal{X}$, 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*

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

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

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

Prop 13 (Markov Chains and Potential Functions)**.** *For $r : \mathcal{X} \to \mathbb{R}$ be a bounded function and for $\beta \in (0,1)$,*

$$R(x) = \mathbb{E}_x \left[\sum_{t=0}^{\infty} \beta^t r(X_t) \right] \tag{1.2}$$

is the unique solution to the equation

$$R(x) = \beta(PR)(x) + r(x), \qquad x \in \mathcal{X}. \tag{1.3}$$

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

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

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

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

Remark 14. • *Notice the expression* (1.3) *can equivalently be written as*

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

• *For the "moreover" part above we can also switch the inequality. You can check in the proof that if $\tilde{R}$ is bounded and such that*

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

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

• *The reward function can be generalized to the form* $r(x, \hat{x})$, *so the next state is included in the reward. Equation* (1.3) *now becomes*

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

Proof. First note that for $R(x)$ in (1.2)

$$R(x) = r(x) + \mathbb{E}_x\Big[\beta\mathbb{E}\Big[\sum_{t=1}^{\infty}\beta^{t-1}r(X_t)\Big|X_1\Big]\Big] = r(x) + \mathbb{E}_x\Big[\beta R(X_1)\Big] = r(x) + \beta(PR)(x)\,.$$

So $R(x)$ is a solution to (1.3).

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

$$\|\hat{R} - R\|_\infty \le \beta \sum_y P_{xy}|R(y) - R(y)| \le \beta\|\hat{R} - R\|_\infty$$

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 $\tilde{R}(x) \ge r(x) + \beta P\tilde{R}(x)$. Repeated substitution gives

$$\begin{aligned}\tilde{R}(x) \ge r(x) + \mathbb{E}_x\left[\beta\tilde{R}(X_1)\right] \ge \ldots \ge \mathbb{E}_x\Big[\sum_{t=0}^{T}\beta^t r(X_t)\Big] + \beta^{T+1}\mathbb{E}_x\Big[\tilde{R}(X_{T+1})\Big] \\ \ge \mathbb{E}_x\Big[\sum_{t=0}^{T}\beta^t r(X_t)\Big] \xrightarrow[T\to\infty]{} R(x).\end{aligned}$$

□

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

Prop 15. *Given a bounded function* $R : \mathcal{X} \to \mathbb{R}$, *we define*

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

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

$$R(x) = \mathbb{E}_x\left[\sum_{t=0}^{\infty}\beta^t r(X_t)\right] \tag{1.4}$$

Conversely, if $R(x)$ *satisfies* (1.4) *then* M_t, $t \in \mathbb{Z}_+$, *is a martingale.*

Proof. If M_t, $t \in \mathbb{Z}_+$, then

$$0 = \mathbb{E}_x\Big[M_t\Big] = R'(X_t) - \mathbb{E}\Big[\beta^T R'(X_T)\Big] - \mathbb{E}\Big[\sum_{t=0}^{T-1} \beta^t r(X_t)\Big]$$

The term $\mathbb{E}\Big[\beta^T R'(X_T)\Big]$ goes to zero, by the Dominated Convergence Theorem. So

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

Conversely, if

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

then by Prop 13, $R(x) = r(x) + \beta\mathbb{E}_x[R(X_1)]$. Applying this gives

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

□

Markov Chain Examples.

The following is an alternative formulation of the previous proposition.

Ex 16. *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 $f : \partial X \to \mathbb{R}_+$ argue that*

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

solves the equation

$$R(x) = (PR)(x) + r(x), \qquad x \notin \partial\mathcal{X} \tag{1.5}$$

$$R(x) = f(x), \qquad x \in \mathcal{X}. \tag{1.6}$$

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

Ex 17 (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 $f : \mathcal{X} \to \mathbb{R}$, 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 [31], Bremaud [10], or Levin & Peres [27]. The analysis of rewards and Markov processes is particularly studied by Doob [14].

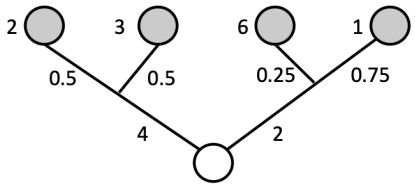
1.3 Markov Decision Processes

- Definition of Markov Decision Process.
- Bellman's Equation.

Markov decision processes are essentially the randomized equivalent of a dynamic program. Let's first consider how to randomize the tree example introduced in Section 1.1.

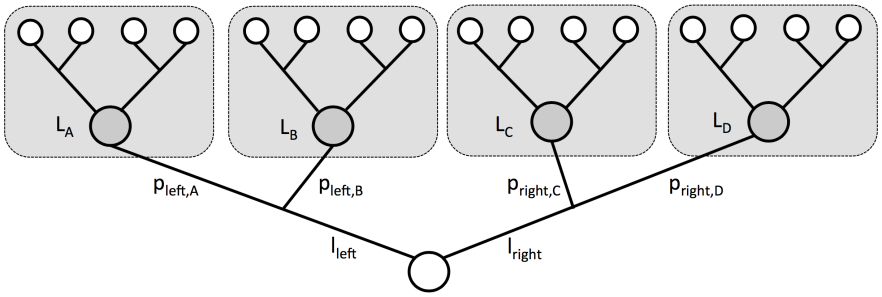
A Random Example

Below is a tree with a root node and four leaf nodes colored grey. At the route 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 & 0.75, and for turning right, the probabilities are 0.25 & 0.75. For each leaf node there is there is a cost, namely, $2, 3, 6,$ and 1.



Given you only know the probabilities (and not what happens when you choose left or right), you'd want to take the decision with lowest expected cost. The expected cost for left is $4 + 0.5 \times 2 + 0.3 \times 3 = 5.5$ and for right is $2 + 0.25 \times 6 + 0.75 \times 1 = 4.25$. So go right.

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

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

where here the random variable X_a denotes the state in $\{A, B, C, D\}$ reached after 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.

Def 18 (Markov Decision Process)**.** *Like with a dynamic program, we consider discrete times $t = 0, 1, ..., T$, states $x \in \mathcal{X}$, actions $a \in \mathcal{A}$ and rewards $r(x, a)$. However, the plant equation and definition of a policy are slightly different. Like with a Markov chain, the state evolves as a random function. Here*

$$X_{t+1} = f(X_t, a_t; U_t) \equiv f(X_t, a_t)$$

for current state X_t and action a_t and where $(U_t)_{t\geq 0}$ are IIDRVs uniform on $[0, 1]$. This is called the Plant Equation.

A policy π choses an action π_t at each time t as a function of past states $x_0, ..., x_t$ and past actions $\pi_0, ..., \pi_{t-1}$. We let $\mathcal{P}$ be the set of policies. A policy, a plant equation, and the resulting sequence of states and rewards describe a Markov Decision Process.

Rmk 19. *As noted in the equivalence above, we will usually suppress dependence on U_t. Also, we will use the notation*

$$\mathbb{E}_{x_t,a_t}[G(X_{t+1})] = \mathbb{E}[G(f(x_t, a_t; U))] \quad \textit{and} \quad \mathbb{E}_{x,a}[G(\hat{X})] = \mathbb{E}[G(f(x, a; U))]$$

where here and here after we use $\hat{X}$ to denote the next state (after taking action a in state x). Notice in both equalities above, the term on the right depends on only one random variable, U.

Objective is to find a process that optimizes the expected reward.

Def 20 (Markov Decision Problem)**.** *Given initial state x_0, a Markov Decision Problem is the following optimization*

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

Further, let $R_\tau(x, \Pi)$ (Resp. $V_\tau(x_\tau)$) be the objective (Resp. optimal objective) for (MDP) *when the summation is started from time $t = T - \tau$ and state $X_{T-\tau} = x$, rather than $t = 0$ and $X_0 = x$. We often call V to value function of the MDP.*

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 21 (Bellman Equation)**.** *Setting* $V_0(x) = r(x)$ *for* $t = 1, ..., T$

$$V_t(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \mathbb{E}_{x,a} \left[V_{t-1}(\hat{X}) \right] \right\}. \qquad \text{(Bell eq.)}$$

This equation is Bellman's equation for a Markov Decision Process.

Proof. Let $\mathcal{P}_t$ be the set policies that can be implemented from time $T-t$ to T. Notice it is the product actions at time t and the set of policies from time $t+1$ onward. [That is $\mathcal{P}_t = \{(\pi_t, \Pi) : \Pi \in \mathcal{P}_{t-1}, \pi_t : \mathcal{X}^{T-t} \times \mathcal{A}^{T-t} \to \mathcal{A}\}$.] So

$$\begin{aligned}
V_t(x) &= \max_{\Pi_t \in \mathcal{P}_t} \mathbb{E}_{x,\pi_{T-t}} \left[\sum_{t=T-t}^{T-1} r(X_t, \pi_t) + r(X_T) \right] \\
&= \max_{\pi_{T-t}} \max_{\Pi \in \mathcal{P}_{t-1}} \left\{ \mathbb{E}_{x_{T-t},\pi_{T-t}} \left[\mathbb{E}_{X_{T-t+1},\pi_{T-t+1}} \left[r(x_{T-t}, \pi_{T-t}) + \sum_{\tau=T-t+1}^{T-1} r(X_\tau, \pi_\tau) + r(X_T) \right] \right] \right\} \\
&= \max_{a \in \mathcal{A}} \left\{ r(x_t, a) + \mathbb{E}_{x,a} \left[\underbrace{\max_{\Pi \in \mathcal{P}_{t-1}} \mathbb{E}_{X_{T-t+1},\pi_{T-t+1}} \left[\sum_{\tau=T-t+1}^{T-1} r(X_\tau, \pi_\tau) + r(X_T) \right]}_{= V_{t-1}(\hat{X})} \right] \right\}
\end{aligned}$$

2nd equality uses structure of $\mathcal{P}_t$, takes the r term out and then takes conditional expectations. 3rd equality takes the supremum over $\mathcal{P}_{t-1}$, which does not depend on π_t, inside the expectation and notes the supremum over π_t is optimized at a fixed action $a \in \mathcal{A}$ (i.e. the past information did not help us.) □

MDP Examples

Ex 22. *You need to sell a car. At every time $t = 0, ..., T-1$, 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 be 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 23 (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 decresing sequence $\{a_t\}_{0 \le t \le T}$ such that it is optimal to exercise whenever $X_s \ge a_s$ occurs.

Ex 24. *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 25. *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 $i \in \{1, ..., n\}$ then either: he esacapes with probability p_p; 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.*

References

Markov decision processes were studied as the natural probabilistic analog of dynamic programs. An early text on MDPs is Howard [21]. Standard texts on dynamic programming [5] or on Markov chains [31] cover MDPs. A good account of MDPs is Puterman [32].

1.4 Infinite Time Horizon

- Discounted Programming, Positive Programming, Negative Programming, Average Programming.
- Conditions for the Optimality of the Bellman Equation.

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

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

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

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

We will cover other cases where $\beta = 1$ later. At this point it is useful 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 27 (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 28. *For a discounted program, the optimal policy $V(x)$ satisfies*

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

Moreover, if we find a function $R(x)$ such that

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

then $R(x) = V(x)$, i.e. the solution to the Bellman equation is unique. Further given such an $R(x)$, if take a function $\pi(x)$ such that

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

Then π is optimal and $R(x,\pi) = R(x) = V(x)$ the optimal value function.

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 a solution is optimal by applying Proposition 13.

We know that $R_t(x,\Pi) = r(x,\pi_0) + \beta \mathbb{E}[R_{t-1}(\hat{X},\hat{\Pi})]$. Applying limits as $t \to \infty$ on both sides and bounded convergence theorem gives that

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

For the inequality, above, we maximize $R(\hat{X},\hat{\Pi})$ over $\hat{\Pi}$. Now maximizing the left hand side over Π gives

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

At this point we have that the Bellman equation 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 have that

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

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 $\epsilon \to 0$ gives

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

Thus we now have that

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

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 this recursion 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

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

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 29 after we complete this proof. Now since $0 < \beta < 1$ the only solution to this inequality 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

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

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 $R(x) = r(x,\pi(x)) + \beta \mathbb{E}_{x,\pi(x)}[R(\hat{X})]$. So by Prop 13, $R(x) = R(x,\pi)$. □

For the above proof we required the following technical lemma.

Lem 29. *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. Clearly it is true that

$$\max_a b(a) + c(a) \leq \max_a b(a) + \max_a c(a)$$

for any two functions $b(a)$ and $c(a)$. Suppose without loss of generality that $\max_a f(a) \geq \max_a g(a)$. If we take $b(a)+c(a) = f(a)$ and $b(a) = g(a)$ in the above expression then

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

Thus, as required,

$$\begin{aligned} |\max_a f(a) - \max_a g(a)| &= \max_a f(a) - \max_a g(a) \\ &\leq \max_a f(a) - g(a) \leq |\max_a f(a) - g(a)|. \end{aligned}$$

□

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

Prop 30. *a) Stationary Q-factors satisfy the recursion*

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

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

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

The optimal value function satisfies

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

c) The operation

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

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

$$\|\boldsymbol{F}(\boldsymbol{Q}_1) - \boldsymbol{F}(\boldsymbol{Q}_2)\|_\infty \le \|\boldsymbol{Q}_1 - \boldsymbol{Q}_2\|_\infty\,.$$

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 $P^a_{x\hat{x}}$. There after it remains in $\mathcal{X}$ moving according to policy π. Thus by Prop 13

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

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

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

Thus

$$\|\boldsymbol{F}(\boldsymbol{Q}_1) - \boldsymbol{F}(\boldsymbol{Q}_2)\|_\infty \leq \beta \max_{\hat{x}} |\max_a Q_1(\hat{x}, a) - \max_{a'} Q_2(\hat{x}, a')| \leq \beta \|\boldsymbol{Q}_1 - \boldsymbol{Q}_2\|_\infty ,$$

as required. □

Positive Programming*

We now consider case were all rewards are positive and the discount factor β can be set equal to 1. This is called *Positive Programming*. The following result holds for positive programming.

Thrm 31. *Consider a positive program the optimal value function $V(x)$ is the minimal non-negative solution to the Bellman equation*

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

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

Proof. 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 21, Bellman's equation holds

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

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

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

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

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

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

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

Thus $R(x) \geq \lim_{T\to\infty} V_T(x) = V(x)$. 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. □

Negative Programming*

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

Def 32. *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 31 for Negative programming is weaker.

Thrm 33. *Consider a negative program, minimizing positive costs. For the minimal non-negative solution to the Bellman equation*

$$L(x) = \min_{a\in\mathcal{A}} \left\{ c(x,a) + \beta \mathbb{E}_{x,a} \left[L(\hat{X}) \right] \right\}, \tag{1.7}$$

any stationary policy Π *that solves the Bellman equation:*

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

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 31: 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

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

and that $\lim L_T(x) = L(x) \leq C(x)$ for any other solution to the Bellman equation. (See the proof of Thrm 31 for more detail.)

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

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

Thus

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

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

Average Programming*

We now consider a slightly different approach to dynamic programming where costs are not discounted. Suppose that for finite-time cost function

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

We look at the limit of the average cost

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

if such a limit exists, and attempt to find the minimal such policy. [We could also maximize rewards if preferred.]

Thrm 34. *If there exists a constant λ and a bounded function $\kappa(x)$ such that*

$$\kappa(x) \le \min_{a\in\mathcal{A}} \Big\{c(x,a) - \lambda + \mathbb{E}_{x,a}[\kappa(\hat{x})]\Big\}. \tag{1.8}$$

Then, for all policies $\tilde{\pi}$,

$$\liminf_{T\to\infty} \frac{C_T(x_0, \tilde{\pi})}{T} \ge \lambda. \tag{1.9}$$

Moreover, if there exists a stationary policy $\pi(x)$ such that

$$\kappa(x) \ge c(x, \pi(x)) - \lambda + \mathbb{E}_{x,\pi(x)}[\kappa(\hat{x})]$$

then

$$\limsup_{T\to\infty} \frac{C_T(x_0, \pi)}{T} \le \lambda$$

and thus the policy π has optimal long-run cost.

Proof. Let

$$M_t = \kappa(X_t) + \sum_{\tau=0}^{t-1} \{c(X_\tau, \tilde{\pi}_\tau) - \lambda\}.$$

Under condition (1.8), M_t is a sub-Martingale:

$$\mathbb{E}[M_{t+1} - M_t | X_t = x, \tilde{\pi}_t = a] = \mathbb{E}_{x,a}[\kappa(\hat{x})] - \kappa(x) + c(x,a) - \lambda \ge 0.$$

Thus

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

and so

$$\liminf_{T\to\infty} \frac{C_T(x,\tilde{\pi})}{T} \geq \lambda.x$$

Under condition (1.9), M_t is a super-Martingale when $\tilde{\pi} = \pi$. So

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

and so

$$\limsup_{T\to\infty} \frac{C_T(x,\tilde{\pi})}{T} \leq \lambda.$$

□

Note we can always add a constant to $\kappa(x)$ in the above theorem. So it is worth specifying that $\kappa(x_0) = 0$ and then we think of $\kappa(x)$ as the additional cost for starting in state x instead of state x_0.

A Martingale Principle of Optimal Control.

We give a Martingale condition for optimal control. This result is analogous to Prop 15 which applies to Markov chains.

Prop 35 (A Martingale Principle of Optimal Control.)**.** *Consider discounted program. Suppose for a bounded function $R : \mathcal{X} \to \mathbb{R}$ 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

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

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

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

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

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

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 $R(x) = V(x) = R(x, \pi)$. □

Infinite Time Examples

Ex 36 (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 37 (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*

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

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 38 (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*

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

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. The are repeatedly arrested by the police and interviewed. Each time they can chose 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 $t = 0, 1, ..., T$ *then it is in the interested 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 confesses 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 $t = 0, 1, 2, 3, ...$ *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 [7, 8] and Staunch [37]. Average reward is considered by Howard [21]. The distinction between these cases is quite standard in different textbooks for instance see [32]. A early review of martingale conditions for optimal control is by Davis [13].

1.5 Algorithms for MDPs

- Policy Improvement; Policy Evaluation.
- Value Iteration; Policy Iteration.
- Temporal Differences; Q-factors.

For infinite time MDPs, we cannot apply to induction on Bellman's equation from some initial state – like we could for finite time MDP. So we need some algorithms to solve MDPs.

At a high level, for a Markov Decision Processes (where the transitions P^a_{xy} 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 new policy π, for instance by solving Bellman's equation:
$$\pi(x) \in \underset{a\in\mathcal{A}}{\operatorname{argmax}} \left\{ r(x,a) + \beta \mathbb{E}_{x,a}\left[R(\hat{X}, \pi_0) \right] \right\}$$
- (Policy Evaluation) Here you find the value of your policy. For instance by finding the reward function for policy π:
$$R(x,\pi) = \mathbb{E}^{\pi}_{x}\left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi(X_t)) \right]$$

Value iteration

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

Def 39 (Value iteration)**.** *Take* $V_0(x) = 0\ \forall x$ *and recursively calculate*

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

for $s = 1, 2, ..$ *this is called* value iteration.

We can think of the two display equations above, respectively, as the policy improvement and policy evaluation steps. Notice, that we don't really need to do the policy improvement step to do each iteration. Notice the policy evaluation step evalutes one action under the new policy π afterwards the value is $V_s(\hat{X})$.

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

Thrm 40. *For positive programming, i.e. where all rewards are positive and the discount factor β belongs to the interval $(0, 1]$, then*

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

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 1. *For reward function $R(x)$ define*

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

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

Proof. Clearly,

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

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

Proof of Thrm 40. Note that $V_1(x) = \max_a r(x,a) \geq 0 = V_0(x)$. Now, since $V_{s+1}(x) = \mathcal{L}V_s(x)$, repeatedly applying Lemma 1 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.

Next note that $V_s(\cdot)$ is the optimal value function for the finite time MDP with rewards $r(x,a)$ and duration s. 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. □

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 probablity 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 $\beta \in (0,1)$.

Def 41 (Policy Iteration)**.** *Given the stationary policy* Π*, we may define a new (improved) stationary policy,* $\mathcal{I}\Pi$*, by choosing for each* x *the action* $\mathcal{I}\Pi(x)$ *that solves the following maximization*

$$\mathcal{I}\Pi(x) \in \operatorname*{argmax}_{a\in\mathcal{A}} \; r(x,a) + \beta\mathbb{E}_{x,a}\left[R(\hat{X},\Pi)\right]$$

where $R(x,\Pi)$ *is the value function for policy* Π*. We then calculate* $R(x,\mathcal{I}\Pi)$*. Policy iteration is the algorithm that takes*

$$\Pi_{n+1} = \mathcal{I}\Pi_n$$

Starting from a stationary policy Π_0*.*

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

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

$$R = r + \beta PR$$

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

Thrm 42. *Under Policy Iteration*

$$R(x,\Pi_{n+1}) \geq R(x,\Pi_n)$$

and, for bounded programming,

$$R(x,\Pi_n) \nearrow V(x) \qquad \textit{as} \quad n \to \infty$$

Proof. By the optimality of $\mathcal{I}\Pi$ with respect to Π we have

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

Thus from the last part of Thrm 13, we know that $R(x,\Pi) \leq R(x,\mathcal{I}\Pi)$. This show that Policy iteration improves solutions. Now we must show it improves to the optimal solution.

First note that

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

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

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

where $\pi^*(x)$ is the optimal policy.[2] To see taking expectations with respect to the optimal policy π^* gives

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

Since M_t is a supermartingale:

$$R(x,\Pi_T) = \mathbb{E}^*_x[M_0] \geq \mathbb{E}^*_x[M_T] = \underbrace{\mathbb{E}^*_x\left[\beta^T R(X_T,\Pi_0)\right]}_{\xrightarrow[T\to\infty]{} 0} + \underbrace{R_T(x,\Pi^*)}_{\xrightarrow[T\to\infty]{} V(x)}$$

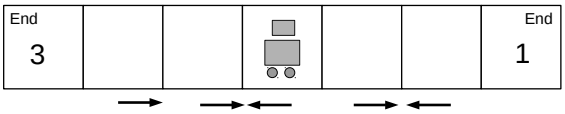
Therefore, as required, $\lim_{T\to\infty} R(x,\Pi_T) \geq V(x)$. □

[2]Note we are implicity 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.

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

    # Arguments:
        P - P[a][x][y] gives probablity 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 - disount 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
```

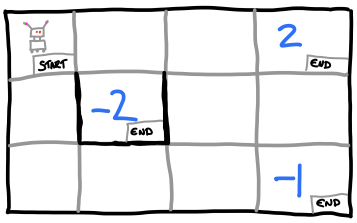
MDP Algorithms – Examples

Ex 43. *Apply the policy iteration algorithm it to the following problem:*



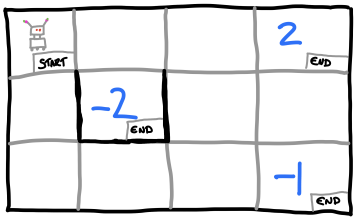
Here a robot must navigate to either end point. It receives a reward of 3 *for reaching the lefthand side and* 1 *for the righthand side. An initial policy from which you must start is provided by the arrows above.*

Ex 44 (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 45 (GridWorld, again)**.** *Write a program that uses, Policy iteration to find the optimal policy for the robot in [44].*



Ex 46. *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 47. *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 [21]. Both are now standard text book methods [5, 32].

1.6 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

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

This defines a *stopping problem*.

Assuming that time is finite, the Bellman equation is

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

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

Def 48 (OLSA rule)**.** *In the one step lookahead (OSLA) rule we stop when ever $x \in S$ where*

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

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

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

Prop 50. *If, for the finite time stopping problem, the set S 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 otherwords $C_{s-1}(\hat{X}) = k(\hat{X})$. Therefore, in this case, Bellman's equation becomes

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

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 is 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. □

Optimal stopping in infinite time

We now give conditions for the one step look ahead rule to be optimal for infinite time stopping problems.

Prop 51. *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 the optimal policy π stops at time τ 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 policy π but for the s time horizon problem and stop at s if $\tau \geq s$ then

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

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

As before (for the finite time problem), it is no 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 $L_s(x) \to L(x)$, we have that $L(x) = k(x)$ for all $x \in S$ and there for it is optimal to stop for $x \in S$. □

Stopping a Random Walk

The one step lookahead rule is not always the correct solution to an optimal stopping problem.

Def 52 (Concave Majorant)**.** *For a function $r : \{0, ..., N\} \to \mathbb{R}_+$ 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 53 (Stopping a Random Walk)**.** *Let X_t be a symmetric random walk on $\{0, ..., N\}$ where the process is automatically stopped at 0 and N. For each $x \in \{0, ..., N\}$, 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, and that it is optimal to stop whenever $V(x) = r(x)$.

Proof. The Bellman equation is

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

with $R(x) = r(0)$ and $R(N) = r(N)$. 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 on 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

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

Since value iteration converges $R_s(x) \nearrow V(x)$, where $V(x)$ satisfies $V(x) \leq G(x)$, as required.

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

Exercises

Ex 54 (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: $1, 2, ..., N$ And arrive for interview uniformly at random. Find the policy that maximises the probability that you hire the best candidate.*

Ex 55 (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 56. *In a game show a contestant is asked a series of 10 questions. For each question $q = 1, ..., 10$ 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 looses 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 57 (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 tth 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 looses all his money. Find the optimal policy for the burglar's retirement.*

Ex 58 (Bruss' Odds Algorithm). *You sequentially treat patients $t = 1, ..., T$ 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

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

This procedure is called Bruss' Odds Algorithm.

Ex 59. *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 60. *You own a "toxic" asset its value, x_t at time t, belongs to $\{1,2,3,...\}$. 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 [12]. An authoritative texts mostly focusing on stopping diffusions is Shiryaev [35]. Ferguson provides a good step of unpublished notes on his website.[3] Again most standard texts on stochastic control cover optimal stopping, see for example Whittle [46].

[3] https://www.math.ucla.edu/~tom/Stopping/Contents.html

1.7 LQR and the 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 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, we may need to replace the variable state, $\boldsymbol{x}$ with its mean $\boldsymbol{x}$, this is called certainty equivalent control. If noise is Gaussian, then estimating $\boldsymbol{x}$ 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.

Def 61 (Linear Quadratic Regularization)**.** *We consider the following optimization:*

$$\begin{aligned} V_0(\boldsymbol{x}_0) = & \textit{minimize} && \sum_{t=0}^{T-1} \{\boldsymbol{x}_t^\top R \boldsymbol{x}_t + \boldsymbol{a}_t^\top Q \boldsymbol{a}_t\} + \boldsymbol{x}_T^\top R \boldsymbol{x}_T \qquad \text{(LQR)} \\ & \textit{subject to} && \boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{a}_{t-1}, \quad t = 1, ..., T \\ & \textit{over} && \boldsymbol{a}_0, ..., \boldsymbol{a}_{T-1} \end{aligned}$$

Here the actions $\boldsymbol{a}$ belong to $\mathbb{R}^m$ and the states $\boldsymbol{x}$ belong to $\mathbb{R}^n$. Here A and B are matrices and R and Q are positive semi-definite matrices.[4]

The objective above 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).t

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:

[4]Recall, a matrix M is positive semi-definite if $\boldsymbol{x}^\top M \boldsymbol{x} > 0$ for $x \neq 0$.

Def 62 (Riccati Equation and Gain Matrix)**.** *The Riccati equation is the following matrix recursion*

$$\Lambda_t = R + A^\top \Lambda_{t+1} A - A^\top \Lambda_{t+1} B[Q - B^\top \Lambda_{t+1} B]^{-1} B^\top \Lambda_{t+1} A \qquad \text{(Riccati)}$$

for $t = 0, ..., T-1$ *and with* $\Lambda_T = R$*. The gain matrix is defined to be*

$$G_t = [Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \qquad \text{(Gain)}$$

Essentially $a = -G_t x$ gives the optimal control at time t.

Solution for LQR. We let $V_\tau(\boldsymbol{x})$ be the optimal solution to (LQR), where the summation is started from time $t = \tau$ in state $\boldsymbol{x}_\tau = \boldsymbol{x}$ The following result gives the solution to an LQR problem.

Thrm 63. *The value function for* (LQR) *satisfies*

$$V_t(\boldsymbol{x}) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x}$$

where Λ_t *is the solution to the Riccati Equation, see* (Riccati)*. Moreover, the optimal control action is given by*

$$\boldsymbol{a}_t^\star = -G_t \boldsymbol{x}_t$$

where G *is the Gain Matrix, see* (Gain)*.*

Proof. The Bellman equation is

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

We now argue by induction that $L_t(x) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x}$ for all t. This is certainly true at time T where $L_T(x) = \boldsymbol{x}^\top R \boldsymbol{x}$.

Assuming by induction that $L_t(x) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x}$, we have that

$$\begin{aligned} L_{t-1}(x) &= \min_a \{\boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{a}^\top Q \boldsymbol{a} + (A\boldsymbol{x} + B\boldsymbol{a})^\top \Lambda_t (A\boldsymbol{x} + B\boldsymbol{a})\} \\ &= \min_a \{\boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{a}^\top Q \boldsymbol{a} + \boldsymbol{x}^\top A^\top \Lambda_t A \boldsymbol{x} + 2\boldsymbol{a}^\top B^\top \Lambda_t A \boldsymbol{x} + \boldsymbol{a}^\top B^\top \Lambda_t B \boldsymbol{a}\} \end{aligned}$$

Differentiating the above objective with respect to $\boldsymbol{a}$ and setting equal to zero, minimizes the above objective and gives the condition

$$0 = 2Q\boldsymbol{a} + 2B^\top \Lambda_t A \boldsymbol{x} + 2B^\top \Lambda_t B a$$

This implies that the optimal action is

$$\boldsymbol{a}^\star = -[Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \boldsymbol{x}\,.$$

In other words we see that $\boldsymbol{a}_t^\star = -G_t \boldsymbol{x}_t$, as require above. However, we still need to verify that $V_{t-1}(x) = \boldsymbol{x}^\top \Lambda_{t-1} \boldsymbol{x}$ to complete the induction step. Substituting our expression for $\boldsymbol{a}^\star$ into the above minimization gives.

$$\begin{aligned} L_{t-1}(\boldsymbol{x}) &= \boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{x}^\top A^\top \Lambda_t A \boldsymbol{x} + \boldsymbol{a}^{\star\top} Q \boldsymbol{a}^\star + 2\boldsymbol{a}^{\star\top} B^\top \Lambda_t A \boldsymbol{x} + \boldsymbol{a}^{\star\top} B^\top \Lambda_t B \boldsymbol{a}^\star \\ &= \boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{x}^\top A^\top \Lambda_t A \boldsymbol{x} - \boldsymbol{x}^\top A^\top \Lambda_t B [Q + B^\top \Lambda_t B]^{-1} B^\top \Lambda_t A \boldsymbol{x} \\ &= \boldsymbol{x}^\top \Lambda_{t-1} \boldsymbol{x} \end{aligned}$$

where the last inequality follows by our definition of Λ_{t-1} from the Riccati equation. □

LQR with Noise.

We consider a small variation on the LQR problem. In particular we assume that x_t is randomly perturbed. We consider following optimization:

$$\begin{aligned} L_0(\boldsymbol{x}_0) = \text{minimize} \quad & \sum_{t=0}^{T-1} \{\boldsymbol{x}_t^\top R \boldsymbol{x}_t + \boldsymbol{a}_t^\top Q \boldsymbol{a}_t\} + \boldsymbol{x}_T^\top R \boldsymbol{x}_T \qquad \text{(Noisy LQR)} \\ \text{subject to} \quad & \boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + Ba_{t-1} + \epsilon_{t-1}, \quad t = 1, ..., T \\ \text{over} \quad & \boldsymbol{a}_0, ..., \boldsymbol{a}_{T-1} \end{aligned}$$

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

$$\mathbb{E}[\boldsymbol{\epsilon}_t] = \mathbf{0} \quad \text{and} \quad \mathbb{E}[\boldsymbol{\epsilon}_t^\top \boldsymbol{\epsilon}_t] = N.$$

The next result shows that optimal control remains the same when we add noise, only the value function changes a little bit.

Thrm 64. *The optimal control for the noisy LQR problem is identical to the LQR problem [without noise] in Theorem 63. The value function now has the form*

$$L_t(x) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x} + \gamma_t$$

where $\gamma_{t-1} = tr(\Lambda_t N) + \gamma_t$ *and* $\gamma_T = 0$.

Proof. The Bellman equation is

$$L_{t-1}(x) = \min_a \{\boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{a}^\top Q \boldsymbol{a} + \mathbb{E}[L_t(A\boldsymbol{x} + B\boldsymbol{a} + \boldsymbol{\epsilon})]\} .$$

We now argue by induction that $L_t(x) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x} + \gamma_t$ for all t. This is certainly true at time T where $L_T(x) = \boldsymbol{x}^\top R \boldsymbol{x}$.

Assuming by induction that $L_t(x) = \boldsymbol{x}^\top \Lambda_t \boldsymbol{x} + \gamma_t$, we have that

$$\begin{aligned} L_{t-1}(x) &= \min_a \{\boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{a}^\top Q \boldsymbol{a} + \mathbb{E}[(A\boldsymbol{x} + B\boldsymbol{a} + \boldsymbol{\epsilon})^\top \Lambda_t (A\boldsymbol{x} + B\boldsymbol{a} + \boldsymbol{\epsilon})] + \gamma_t\} \\ &= \min_a \{\boldsymbol{x}^\top R \boldsymbol{x} + \boldsymbol{a}^\top Q \boldsymbol{a} + \boldsymbol{x}^\top A^\top \Lambda_t A \boldsymbol{x} + 2\boldsymbol{a}^\top B^\top \Lambda_t A \boldsymbol{x} + \boldsymbol{a}^\top B^\top \Lambda_t B \boldsymbol{a}\} \\ &\quad + \mathbb{E}[\boldsymbol{\epsilon}^\top \Lambda_t \boldsymbol{\epsilon}] + \gamma_t \end{aligned}$$

First, observe the minimization above is identical to the LQR minimization in Theorem 63, and so equals $\boldsymbol{x}\Lambda_{t-1}\boldsymbol{x}$ by the proof given in Theorem 63. Second observe, a quick calculation shows that

$$\mathbb{E}[\boldsymbol{\epsilon}^\top \Lambda_t \boldsymbol{\epsilon}] = \sum_{ij} \mathbb{E}[\epsilon_i \epsilon_j \Lambda_{t,ij}] = tr(N\Lambda).$$

Thus $\gamma_t + \mathbb{E}[\boldsymbol{\epsilon}^\top \Lambda_t \boldsymbol{\epsilon}] = \gamma_{t-1}$ as defined above. These two observations give that

$$L_{t-1}(x) = \boldsymbol{x}^\top \Lambda_{t-1} \boldsymbol{x} + \gamma_{t-1}$$

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 $\boldsymbol{x}$ but instead some measurement $\boldsymbol{y}$ which we must use to control $\boldsymbol{x}$. Further both $\boldsymbol{x}$ and $\boldsymbol{y}$ are subject to noise.

$$\begin{aligned} L_0(\boldsymbol{x}_0) = \text{minimize} \quad & \sum_{t=0}^{T-1} \{\boldsymbol{x}_t^\top R \boldsymbol{x}_t + \boldsymbol{a}_t^\top Q \boldsymbol{a}_t\} + \boldsymbol{x}_T^\top R \boldsymbol{x}_T \qquad \text{(LQG)} \\ \text{subject to} \quad & \boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{a}_{t-1} + \boldsymbol{\epsilon}_{t-1}, \\ & \boldsymbol{y}_t = C\boldsymbol{x}_{t-1} + \boldsymbol{\delta}_{t-1} \quad t = 1, ..., T \\ \text{over} \quad & \boldsymbol{a}_0, ..., \boldsymbol{a}_{T-1} \end{aligned}$$

In addition to terms in 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

$$\mathbb{E}[\boldsymbol{\delta}] = 0, \qquad \text{and} \qquad \mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top] = M.$$

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 yet.

Here the state $\boldsymbol{x}$ is not directly observed. So we must base decisions from data of past decision and measurements, that is

$$F_t = (\boldsymbol{y}_t, ..., \boldsymbol{y}_1, \boldsymbol{a}_{t-1}, ..., \boldsymbol{a}_0)\,.$$

Result on LQG. The key result on LQG is that if we can estimate the mean state given F_t, i.e. $\boldsymbol{x}_t := \mathbb{E}[\boldsymbol{x}|F_t]$, then the optimal control is that same as from the LQR problem i.e. $a^\star = -G\bar{x}_t$.

Thrm 65. *For an LQG problem the optimal control at time t is*

$$a_t^\star = -G_t\bar{\boldsymbol{x}}_t$$

where $\bar{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_t|F_t]$ and G_t is (Gain)*. Further, the optimal value function is*

$$L_t(F_t) = \mathbb{E}[\boldsymbol{x}_t^\top \Lambda_t \boldsymbol{x}_t|F_t] + I_t + \gamma_t$$

where

$$I_t = \sum_{\tau=t}^{T-1} \mathbb{E}\Big[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau)\Delta_\tau|F_t\Big] \qquad \textit{and} \qquad \Delta_t = \boldsymbol{x}_t - \bar{\boldsymbol{x}}_t\,.$$

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, bit 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 and states 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 2. *Δ_τ is a constant with respect to $\boldsymbol{a}_0, ..., \boldsymbol{a}_\tau$.*

Proof. We recursively consider the update equation for $\boldsymbol{x}_\tau$. Note that

$$\begin{aligned}
\boldsymbol{x}_\tau &= A\boldsymbol{x}_{\tau-1} + B\boldsymbol{a}_{\tau-1} + \boldsymbol{\epsilon}_{\tau-1} \\
&= A[A\boldsymbol{x}_{\tau-2} + B\boldsymbol{a}_{\tau-2} + \boldsymbol{\epsilon}_{\tau-2}] + B\boldsymbol{a}_{\tau-1} + \boldsymbol{\epsilon}_{\tau-1} \\
&= A^2\boldsymbol{x}_{\tau-2} + AB\boldsymbol{a}_{\tau-2} + B\boldsymbol{a}_{\tau-1} + A\boldsymbol{\epsilon}_{\tau-2} + \boldsymbol{\epsilon}_{\tau-1} \\
&\vdots \\
&= A^\tau \boldsymbol{x}_0 + \sum_{t=0}^{\tau-1} A^{\tau-1-t} B\boldsymbol{a}_t + \sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t \,.
\end{aligned}$$

Consequently notice,

$$\bar{\boldsymbol{x}}_\tau = \mathbb{E}[\boldsymbol{x}_\tau | F_\tau] = A^\tau \boldsymbol{x}_0 + \sum_{t=0}^{\tau-1} A^{\tau-1-t} B\boldsymbol{a}_t + \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t \Big| F_\tau\Big]$$

So

$$\boldsymbol{x}_\tau - \bar{\boldsymbol{x}}_\tau = \sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t - \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t \Big| F_\tau\Big].$$

It seems like we are done, we have removed all dependence on the actions taken. But remember $F_t = (\boldsymbol{y}_t, ..., \boldsymbol{y}_1, \boldsymbol{a}_{t-1}, ..., \boldsymbol{a}_0)$ which we condition on above. In principle, we could modify the set of actions that we take to infer information about $\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t$ and thus there would be dependence on the actions taken in the conditional expectation above. However, this turns out not to be the case.

To see this, first, let $\boldsymbol{y}_t^0$ be the sequence of observations made when actions are chosen to be zero, that is, since

$$\boldsymbol{y}_t = C\boldsymbol{x}_t = CA^\tau \boldsymbol{x}_0 + \sum_{t=0}^{\tau-1} CA^{\tau-1-t} B\boldsymbol{a}_t + \sum_{t=0}^{\tau-1} CA^{\tau-1-t}\boldsymbol{\epsilon}_t$$

then $\boldsymbol{y}_t^0$ is given by

$$\boldsymbol{y}_t^0 = CA^\tau \boldsymbol{x}_0 + \sum_{t=0}^{\tau-1} CA^{\tau-1-t}\boldsymbol{\epsilon}_t + \boldsymbol{\delta}_t$$

Since we know which actions we take we can always construct $\boldsymbol{y}_t^0$ from $\boldsymbol{y}_t$ and vice versa. I.e. conditioning on $F_t = (\boldsymbol{y}_t, ..., \boldsymbol{y}_1, \boldsymbol{a}_{t-1}, ..., \boldsymbol{a}_0)$ is the same as conditioning on $F_t^0 = (\boldsymbol{y}_t^0, ..., \boldsymbol{y}_1^0, \boldsymbol{a}_{t-1}, ..., \boldsymbol{a}_0)$. Further,

suppose we let π be the policy that we use to select the actions. In particular, suppose that $\boldsymbol{a}_t = \pi(F_t)$ 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 $\boldsymbol{a}_t = \pi^0(F_t^0)$ 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 $(\boldsymbol{y}_t^0, ..., \boldsymbol{y}_1^0)$. In other words, conditioning on F_t^0 is the same as conditioning on $(\boldsymbol{y}_t^0, ..., \boldsymbol{y}_1^0)$ and the deterministic function π^0. However, π^0 is deterministic and thus independent of the random variable $\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t$, thus it plays no role in determining its conditional expectation. In summary we have found that

$$\begin{aligned}\mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t\Big|F_\tau\Big] &= \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t\Big|F_\tau^0\Big] \\ &= \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t\Big|\boldsymbol{y}_t^0, ..., \boldsymbol{y}_1^0, \pi^0\Big] = \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t\Big|\boldsymbol{y}_t^0, ..., \boldsymbol{y}_1^0\Big]\end{aligned}$$

The right hand expression does not depend on the action taken and so the same is true of

$$\boldsymbol{x}_\tau - \bar{\boldsymbol{x}}_\tau = \sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t - \mathbb{E}\Big[\sum_{t=0}^{\tau-1} A^{\tau-1-t}\boldsymbol{\epsilon}_t\Big|F_\tau\Big].$$

□

A further slightly more minor observation is the following Lemma.

Lemma 3.

$$\mathbb{E}[\boldsymbol{x}_t^\top M\boldsymbol{x}_t|F_t] = \bar{\boldsymbol{x}}_t^\top M\bar{\boldsymbol{x}}_t - \mathbb{E}[\Delta_t^\top M\Delta_t|F_t]$$

Proof.

$$\begin{aligned}\mathbb{E}[\boldsymbol{x}_t^\top M\boldsymbol{x}_t|F_t] &= \mathbb{E}[(\bar{\boldsymbol{x}}_t + \Delta)^\top M(\bar{\boldsymbol{x}}_t + \Delta)|F_t] \\ &= \bar{\boldsymbol{x}}_t M\bar{\boldsymbol{x}}_t + 2\bar{\boldsymbol{x}}_t M\mathbb{E}[\Delta_t|F_t] + \mathbb{E}[\boldsymbol{x}_t^\top M\boldsymbol{x}_t|F_t] \\ &= \bar{\boldsymbol{x}}_t^\top M\bar{\boldsymbol{x}}_t - \mathbb{E}[\Delta_t^\top M\Delta_t|F_t]\end{aligned}$$

□

Proof of Theorem 65 We can now use the above lemma to prove Theorem 65.

Proof of Theorem 65. The result of the Theorem is certainly true at time T, where $L(F_T) = \boldsymbol{x}_T^\top R \boldsymbol{x}_T$. Let's work back inductively assuming the form $L_{t+1}(F_{t+1})$ holds to prove the result.

$$\begin{aligned} L_t(F_t) &= \min_{a_t} \mathbb{E}[\boldsymbol{x}_t^\top R \boldsymbol{x}_t + \boldsymbol{a}_t^\top Q \boldsymbol{a}_t + L_{t+1}(F_{t+1})|F_t] \\ &= \min_{a_t} \mathbb{E}\Big[\underbrace{\boldsymbol{x}_t^\top R \boldsymbol{x}_t}_{(a)} + \boldsymbol{a}_t^\top Q \boldsymbol{a}_t + \underbrace{\mathbb{E}\Big[\boldsymbol{x}_{t+1}^\top \Lambda_{t+1} \boldsymbol{x}_{t+1} | F_{t+1}\Big]}_{(b)} + \underbrace{I_{t+1} + \gamma_{t+1}}_{(c)} \Big| F_t \Big] \end{aligned}$$

Let's deal with the three terms (a), (b) and (c) above.

Firstly, for (a) we have by Lemma 3 that

$$\mathbb{E}[\boldsymbol{x}_t^\top R \boldsymbol{x}_t | F_t] = \bar{\boldsymbol{x}}_t^\top R \bar{\boldsymbol{x}}_t + \mathbb{E}[\Delta_t^\top R \Delta_t | F_t]$$

Second, for term (b):

$$\begin{aligned} &\mathbb{E}\Big[\boldsymbol{x}_{t+1}^\top \Lambda_{t+1} \boldsymbol{x}_{t+1} \Big| F_{t+1}\Big] \\ &= \mathbb{E}\Big[(A\boldsymbol{x}_t + B\boldsymbol{a}_t + \boldsymbol{\epsilon}_t)^\top \Lambda_{t+1} (A\boldsymbol{x}_t + B\boldsymbol{a}_t + \boldsymbol{\epsilon}_t) \Big| F_{t+1}\Big] \\ &= \mathbb{E}[\boldsymbol{x}_t^\top A^\top \Lambda_{t+1} A \boldsymbol{x}_t | F_t] + 2\bar{\boldsymbol{x}}_t A^\top \Lambda_{t+1} B a_t + \boldsymbol{a} B^\top \Lambda_{t+1} B \boldsymbol{a} + tr(N\Lambda_{t+1}) \\ &= \bar{\boldsymbol{x}}_t^\top A^\top \Lambda_{t+1} A \bar{\boldsymbol{x}}_t + \mathbb{E}[\Delta_t^\top A^\top \Lambda_{t+1} A \Delta_t | F_t] \\ &\quad + 2\bar{\boldsymbol{x}}_t A^\top \Lambda_{t+1} B a_t + \boldsymbol{a} B^\top \Lambda_{t+1} B \boldsymbol{a} + tr(N\Lambda_{t+1}) \end{aligned}$$

Third, for term (c),

$$\begin{aligned} \mathbb{E}[I_{t+1} + \gamma_{t+1} | F_t] &= \sum_{\tau=t+1}^{T-1} \mathbb{E}\left[\mathbb{E}\Big[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau \Big| F_{t+1}\Big] \Big| F_t \right] + \gamma_{t+1} \\ &= \sum_{\tau=t+1}^{T-1} \mathbb{E}\left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau \Big| F_t \right] + \gamma_{t+1} \,. \end{aligned}$$

Applying the last three terms to $L_t(F_t)$, above, we get that

$$\begin{aligned} &L_t(F_t) \\ &= \min_a \{\bar{\boldsymbol{x}}_t^\top R \bar{\boldsymbol{x}}_t + \boldsymbol{a}^\top Q \boldsymbol{a} + \bar{\boldsymbol{x}}_t^\top A^\top \Lambda_{t+1} A \bar{\boldsymbol{x}}_t + 2\boldsymbol{a}^\top B^\top \Lambda_{t+1} A \bar{\boldsymbol{x}}_t + \boldsymbol{a}^\top B^\top \Lambda_{t+1} B \boldsymbol{a}\} \\ &\quad + \mathbb{E}[\Delta_t^\top [R + A^\top \Lambda_{t+1} A] \Delta_t | F_t] + \sum_{\tau=t+1}^{T-1} \mathbb{E}\left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau \Big| F_t \right] \\ &\quad + tr(N\Lambda_{t+1}) + \gamma_{t+1} \end{aligned}$$

Critically, we have applied Lemma 2, 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 $a_t^\star = -G\bar{\boldsymbol{x}}_t$, by the same calculation done in Theorem 63. So it is equal to $\bar{\boldsymbol{x}}_t^\top \Lambda_t \bar{\boldsymbol{x}}_t$. So applying this and Lemma , i.e. that $\bar{\boldsymbol{x}}_t \Lambda_t \bar{\boldsymbol{x}}_t = \mathbb{E}[\boldsymbol{x}_t^\top \Lambda_t \boldsymbol{x}_t | F_t] - \mathbb{E}[\Delta_t^\top \Lambda_t \Delta_t | F_t]$. This gives that

$$\begin{aligned}
&L_t(F_t) \\
&= \mathbb{E}[\boldsymbol{x}_t^\top \Lambda_t \boldsymbol{x}_t | F_t] \\
&\quad + \mathbb{E}[\Delta_t^\top [R + A^\top \Lambda_{t+1} A - \Lambda_t] \Delta_t | F_t] + \sum_{\tau=t+1}^{T-1} \mathbb{E}\left[\Delta_\tau^\top (R + A^\top \Lambda_{\tau+1} A - \Lambda_\tau) \Delta_\tau \middle| F_t \right] \\
&\quad + tr(N\Lambda_{t+1}) + \gamma_{t+1} \\
&= \mathbb{E}[\boldsymbol{x}_t^\top \Lambda_t \boldsymbol{x}_t | F_t] + I_t + \gamma_t \,,
\end{aligned}$$

where we apply the definitions of I_t and γ_t. This gives the required expression of $L_t(F_t)$. □

Kalman Filter

Kalman filtering (and filtering in general) considers the following setting: we have a sequence of states $\boldsymbol{x}_t$, which evolves under random perturbations over time. Unfortunately we cannot observe $\boldsymbol{x}_t$, we can only observe some noisy function of $\boldsymbol{x}_t$, namely, $\boldsymbol{y}_t$. Our task is to find the best estimate of $\boldsymbol{x}_t$ given our observations of $\boldsymbol{y}_t$.

Consider the equations

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= A\boldsymbol{x}_t + B\boldsymbol{a}_t + \boldsymbol{\epsilon}_t \\
\boldsymbol{y}_{t+1} &= C\boldsymbol{x}_{t+1} + \boldsymbol{\delta}_{t+1} \,.
\end{aligned}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma^\epsilon)$, $\boldsymbol{\delta}_{t+1} \sim \mathcal{N}(\mathbf{0}, \Sigma^\delta)$ 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

$$\bar{\boldsymbol{x}}_{t+1|t} = A\bar{\boldsymbol{x}}_{t+1|t} + Ba_t \qquad \text{(Predict-1)}$$
$$P_{t+1|t} = AP_{t|t}A^\top + \Sigma_t^\epsilon \qquad \text{(Predict-2)}$$

and

$$\bar{\boldsymbol{x}}_{t+1} = \bar{\boldsymbol{x}}_{t+1|t} + K_t(\boldsymbol{y}_{t+1} - C\bar{\boldsymbol{x}}_{t+1|t}) \qquad \text{(Measure-1)}$$
$$P_{t+1} = P_{t+1|t} - K_t C P_{t+1|t} \qquad \text{(Measure-2)}$$

where

$$K_t = P_{t+1|t}C^\top(CP_{t+1|t}C^\top + \Sigma_t^\delta)\,.$$

The matrix K_t is often referred to as the Kalman Gain. Assuming the initial state $\boldsymbol{x}_0$ 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 66. *Let $\boldsymbol{u}$ be normally distributed vector with mean $\bar{\boldsymbol{u}}$ and covariance $\Sigma_{\boldsymbol{u}}$, i.e.*

$$\boldsymbol{u} \sim \mathcal{N}(\bar{\boldsymbol{u}}, \Sigma_{\boldsymbol{u}})\,.$$

i) For any matrix A and (constant) vector $\boldsymbol{c}$, we have that

$$A\boldsymbol{u} + \boldsymbol{c} \sim \mathcal{N}(A\bar{\boldsymbol{u}} + \boldsymbol{c}, A\Sigma_{\boldsymbol{u}}A^\top)\,.$$

ii) If we take $\boldsymbol{u} = (\boldsymbol{v}, \boldsymbol{w})$ then $\boldsymbol{w}$ conditional on $\boldsymbol{v}$ gives

$$(\boldsymbol{w}|\boldsymbol{v}) \sim \mathcal{N}(\bar{\boldsymbol{w}} + \Sigma_{wv}\Sigma_{vv}^{-1}(\boldsymbol{v} - \bar{\boldsymbol{v}}), \Sigma_{ww} - \Sigma_{wv}\Sigma_{vv}^{-1}\Sigma_{vw})$$

iii) $Var(A\boldsymbol{u}) = A\Sigma_u A^\top$, $Cov(A\boldsymbol{u}, B\boldsymbol{u}) = A\Sigma_u B^\top$.

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 67.

$$[\boldsymbol{x}_{t+1}|\boldsymbol{y}_{[0:t]}, \boldsymbol{a}_{[0:t]}] \sim \mathcal{N}(\bar{\boldsymbol{x}}_{t+1|t}, P_{t+1|t})$$
$$[\boldsymbol{x}_{t+1}|\boldsymbol{y}_{[0:t+1]}, \boldsymbol{a}_{[0:t]}] \sim \mathcal{N}(\bar{\boldsymbol{x}}_{t+1}, P_{t+1})$$

where $\boldsymbol{y}_{[0:t]} := (\boldsymbol{y}_0, ..., \boldsymbol{y}_t)$ and $\boldsymbol{a}_{[0:t]} := (\boldsymbol{a}_0, ..., \boldsymbol{a}_t)$. Thus

$$\mathbb{E}[\bar{x}_{t+1}|F_{t+1}] = \bar{\boldsymbol{x}}_{t+1}$$

where $\bar{\boldsymbol{x}}_{t+1}$ is given by (Measure-1).

Proof. We show the result by induction supposing that

$$[\boldsymbol{x}_t|\boldsymbol{y}_{[0:t]}, \boldsymbol{a}_{[0:t-1]}] \sim \mathcal{N}(\bar{\boldsymbol{x}}_t, P_t)\,.$$

Since $\boldsymbol{x}_{t+1}$ is a linear function of $\boldsymbol{x}_t$, we have that

$$[\boldsymbol{x}_{t+1}|\boldsymbol{y}_{[0:t]}, \boldsymbol{a}_{[0:t]}] \sim \mathcal{N}(\bar{\boldsymbol{x}}_{t+1|t}, P_{t+1|t}) .$$

where, by Prop 66ii), we have that

$$\bar{\boldsymbol{x}}_{t+1|t} = A\bar{\boldsymbol{x}}_t + Ba_t , \qquad P_{t+1|t} = AP_tA^\top + \Sigma^\epsilon .$$

Given $\boldsymbol{y}_{t+1} = C\boldsymbol{x}_{t+1} + \boldsymbol{\delta}_t$, we have by Prop 66iii) that $Var(\boldsymbol{y}_{t+1}|\boldsymbol{y}_{[0,t]}, \boldsymbol{a}_{[0,t]}) = CP_{t+1|t}C^\top$ and $Cov(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}|\boldsymbol{y}_{[0,t]}, \boldsymbol{a}_{[0,t]}) = P_{t+1|t}C^\top$. Thus

$$[(\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1})|\boldsymbol{y}_{[0:t]}, \boldsymbol{a}_{[0:t]}] \sim \mathcal{N}\left([\bar{\boldsymbol{x}}_{t+1|t}, C\bar{\boldsymbol{x}}_{t+1|t}], \begin{bmatrix} P_{t+1|t} & P_{t+1}C^\top \\ CP_{t+1|t} & CP_{t+1}C^\top + \Sigma_t^\delta \end{bmatrix}\right) .$$

Thus applying Prop 66ii), we get that

$$\begin{aligned}[x_{t+1}|y_{[0:t+1]}, a_{[0:t]}] &= [[x_{t+1}|y_{[0:t]}, a_{[0:t]}]|y_{t+1}] \\ &\sim \mathcal{N}\Big(\bar{\boldsymbol{x}}_{t+1|t} + P_{t+1|t}C^\top[CP_{t+1}C^\top + \Sigma_t^\delta]^{-1}(y_{t+1} - C\bar{\boldsymbol{x}}_{t+1|t}) , \\ &\qquad P_{t+1|t} - P_{t+1|t}C^\top[CP_{t+1|t}C^\top + \Sigma_t^\delta]^{-1}CP_{t+1|t}\Big) .\end{aligned}$$

That is, as required, $[\boldsymbol{x}_{t+1}|\boldsymbol{y}_{[0:t+1]}, \boldsymbol{a}_{[0:t]}] \sim \mathcal{N}(\bar{\boldsymbol{x}}_{t+1}, P_{t+1})$ for

$$\begin{aligned}\bar{\boldsymbol{x}}_{t+1} &= \bar{\boldsymbol{x}}_{t+1\,|\,t} + K_t(\boldsymbol{y}_{t+1} - C\bar{\boldsymbol{x}}_{t+1\,|\,t}) \\ P_{t+1} &= P_{t+1\,|\,t} - K_tCP_{t+1\,|\,t}\end{aligned}$$

where $K_t = P_{t+1\,|\,t}C^\top(CP_{t+1|t}C^\top + \Sigma_t^\delta)$. □

References

Bucy and Kalman developed the Kalman filter [11]. It is used extensively in control theory, for a recent text see Grenwal and Andrews [18]. For a machine learning and Bayesian perspective see Murphy [30].

Chapter 2

Continuous Time Control

2.1 Continuous Time Dynamic Programming

- The Hamilton-Jacobo-Bellman equation; a heuristic derivation; and proof of optimality.
- Linear Quadratic Regularization.

Discrete time Dynamic Programming was given in Section 1.1. 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 $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$, the state evolves according to a differential equation

$$\frac{dx_t}{dt} = f(x_t, a_t). \tag{2.1}$$

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

$$\begin{aligned} L(x_0) := \textit{Minimize} \quad & C(\mathbf{a}) := \int_0^T e^{-\alpha t} c(x_t, a_t) dt + e^{-\alpha T} c(x_T) && \text{(DP)} \\ \textit{subject to} \quad & \frac{dx_t}{dt} = f(x_t, a_t), && t \in \mathbb{R}_+ \\ \textit{over} \quad & a_t \in \mathcal{A}, && t \in \mathbb{R}_+ \end{aligned}$$

Further, let $C_\tau(\mathbf{a})$ (Resp. $L_\tau(x_\tau)$) be the objective (Resp. optimal objective) for (2.1) *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 69 (Hamilton-Jacobi-Bellman Equation)**.** *For a continuous-time dynamic program* (2.1)*, the equation*

$$0 = \min_{a \in \mathcal{A}} \{c(x,a) + \partial_t L_t(x) + f(x,a)\partial_x L_t(x) - \alpha L_t(x).\} \tag{HJB}$$

is called the Hamilton-Jacobi-Bellman equation. It is the continuous time analogoue of the Bellman equation [2].

A Heuristic Derivation of the HJB Equation

We now argue why the Hamiliton-Jacobi-Bellman equation is a good candidate for the Bellman equation in continuous time.

A good approximation to the plant equation (2.1) is

$$x_{t+\delta} - x_t = \delta f(x_t, a_t) \tag{2.2}$$

for $\delta > 0$ small, and a good approximation for the objective is

$$C(\mathbf{a}) := \sum_{t \in \{0,\delta,\ldots,(T-\delta)\}} (1-\alpha\delta)^{t/\delta} c(x_t, a_t)\delta + (1-\alpha\delta)^{t/\delta} c(x_T) \tag{2.3}$$

This follows from the definition of the Riemann Integral and we further use the fact that $(1-\alpha\delta)^{t/\delta} \to e^{-\alpha t}$ as $\delta \to 0$.

The Bellman equation for the discrete time dynamic program with objective (2.3) and plant equation (2.2) is

$$L_t(x) = \min_{a \in \mathcal{A}} \{c(x,a)\delta + (1-\alpha\delta)L_{t+\delta}(x_t + \delta f(x,a))\}$$

If we minus $L_t(x)$ from each side in this Bellman equation and then divide by δ and let $\delta \to 0$ 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,

$$\frac{(1-\alpha\delta)L_{t+\delta}(x+\delta f) - L_t(x)}{\delta} \xrightarrow[\delta \to 0]{} \partial_t L_t(x) + f(x,a)\partial_x L_t(x) - \alpha L_t(x).$$

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 70 (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 $C = C_t(\tilde{x}_t,\Pi)$:

$$\begin{aligned}-\frac{d}{dt}\left(e^{-\alpha t}C_t(\tilde{x}_t,\Pi)\right) &= e^{-\alpha t}\left\{c(\tilde{x}_t,\tilde{\pi}_t) - \left[c(\tilde{x}_t,\tilde{\pi}_t) - \alpha C + f(\tilde{x}_t,\tilde{\pi}_t)\partial_x C + \partial_t C\right]\right\}\\ &\leq e^{-\alpha t}c(\tilde{x}_t,\tilde{\pi}_t)\end{aligned}$$

The inequality holds since the term in the square brackets is the objective of the HJB equation, which is *not* maximized by $\tilde{\pi}_t$. □

Linear Quadratic Regularization

Def 71 (LQ problem)**.** *We consider a dynamic program of the form*

$$\begin{aligned} &\textit{Minimize} && \int_0^T [x_t Q x_t + a_t R a_t]\, dt + x_T Q_T x_T && \text{(LQ)} \\ &\textit{subject to} && \frac{dx_t}{dt} = Ax_t + Ba_t, && t \in \mathbb{R}_+ \\ &\textit{over} && a_t \in \mathbb{R}^m, && t \in \mathbb{R}_+. \end{aligned}$$

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 72 (Riccarti Equation)**.** *The differential equation with*

$$\dot{\Lambda}(t) = -Q - \Lambda(t)A - A^\top \Lambda(t) + \Lambda(t)BR^{-1}B^\top \Lambda(t) \quad \textit{and} \quad \Lambda(T) = Q_T. \text{ (RicEq)}$$

is called the Riccarti equation.

Thrm 73. *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 Riccarti 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

$$0 = \min_{a \in \mathbb{R}^n} \left\{ x^\top Q x + a^\top R a + x^\top \dot{\Lambda}(t) x + 2x^\top \Lambda(x)(Ax + Ba) \right\}.$$

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 Riccarti equation has a cost function that solves the Bellman equation and thus by Theorem 70 the policy is optimal. □

2.2 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,

$$\sum_{t\in\{0,\delta,..,T\}} (B_{t+\delta} - B_t) = B_T \quad \text{and} \quad \sum_{t\in\{0,\delta,..,T\}} (B_{t+\delta} - B_t)^2 \approx T \tag{2.4}$$

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

$$\sum_{t\in\{0,\delta,..,T\}} \sigma(X_t)\,(B_{t+\delta} - B_t) \approx \int_0^T \sigma(X_t) dB_t$$

and

$$\sum_{t\in\{0,\delta,..,T\}} \mu(X_t)\,(B_{t+\delta} - B_t)^2 \approx \int_0^T \mu(X_t) dt.$$

The first sum, above, is approximation from a Riemann-Stieltjes integral, i.e.

$$\int_0^T f(t) dg(t) \approx \sum_{t\in\{0,\delta,..,T\}} f(t)(g(t+\delta) - g(t)).$$

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

$$X_{t+\delta} - X_t = \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta, \qquad t = 0, \delta, 2\delta, ... \tag{2.5}$$

then, by summing over values of $t \in \{0, \delta, ..., T - \delta\}$, 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, $\frac{df(x_t)}{dt} = f'(x_t)\frac{dx_t}{dt}$. Ito'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

$$\begin{aligned}
&f(X_{t+\delta}) - f(X_t) \\
=&f(X_t + \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta) - f(X_t) \\
=&f'(X_t)\left\{\mu\delta + \sigma \cdot (B_{t+\delta} - B_t)\right\} + \frac{f''(X_t)}{2}\left\{\mu\delta + \sigma \cdot (B_{t+\delta} - B_t)\right\}^2 + o(\delta) \\
=&f'(X_t)\left\{\mu\delta + \sigma \cdot (B_{t+\delta} - B_t)\right\} + \frac{f''(X_t)}{2}\sigma^2 \cdot (B_{t+\delta} - B_t)^2 + o(\delta)
\end{aligned}$$

In the last equality we use that $(B_{t+\delta} - B_t) = o(\delta^{1/2})$ (which follows from (2.4)). Thus we see that

$$f(X_{t+\delta}) - f(X_t) \approx \left[f'(X_t)\mu(X_t) + \frac{\sigma(X_t)^2}{2}f''(X_t)\right]\delta + f'(X_t)\sigma(X_t)\left(B_{t+\delta} - B_t\right).$$

Consequently we expecrt 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.

2.3 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 1.3.

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 74 (Plant Equation)**.** *Given functions* $\mu_t(X_t, a_t) = (\mu_t^i(X_t, a_t) : i = 1, .., n)$ *and* $\sigma_t(X_t, a_t) = (\sigma_t^{ij}(X_t, a_t) : i = 1, .., n, j = 1, ..., m)$*, 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 75 (Diffusion Control Problem)**.** *Given initial state* x_0*, a dynamic program is the optimization*

$$L(x_0) := \underset{\Pi \in \mathcal{P}}{\text{minimize}}\ C(x_0, \Pi) := \mathbb{E}_{x_0}\left[\int_0^T e^{-\alpha t} c_t(X_t, \pi_t)dt + e^{-\alpha T} c_T(X_T)\right] \quad \text{(DCP)}$$

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 76 (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.[1] *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 1.3 (Theorem 21) and our heuristic derivation of the stochastic integral in Section A.2. This is analogous to continuous time control in Section 2.1.

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 74 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 (2.5)). Similarly the cost function in (DCP) can be approximated by

$$C_t(x, \Pi) \approx \mathbb{E}\left[\sum_{t\in\{0,\delta,\ldots,T-\delta\}} (1-\alpha\delta)^{\frac{t}{\delta}} c_t(X_t, \pi_t)\delta + (1-\alpha\delta)^{\frac{T}{\delta}} c_T(X_T)\right].$$

This follows from the definition of a Riemann Integral and since $(1-\alpha\delta)^{\frac{t}{\delta}} \to e^{-\alpha t}$. The Bellman equation for this objective function and plant equation is satisfies

$$L_t(x) = \min_{a\in\mathcal{A}} \{c_t(x,a)\delta + (1-\alpha\delta)\mathbb{E}_{x,a}\left[L_{t+\delta}(X_{t+\delta})\right]\}.$$

or, equivalently,

$$0 = \min_{a\in\mathcal{A}} \left\{c_t(x,a) + \frac{1}{\delta}\mathbb{E}_{x,a}\left[L_{t+\delta}(X_{t+\delta}) - L_t(x)\right] - \alpha\mathbb{E}_{x,a}\left[L_{t+\delta}(X_{t+\delta})\right]\right\}.$$

[1] Here $[\sigma^\mathsf{T}\sigma]\cdot\partial_{xx}L_t(x)$ is the dot-product of the Hessian matrix $\partial_{xx}L_t(x)$ with $\sigma^\mathsf{T}\sigma$. 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

$$\frac{1}{\delta} \mathbb{E}_{x,a} \left[L_{t+\delta}(X_{t+\delta}) - L_t(x)\right] = \partial_t L + \mu_t(X_t, \pi_t) \cdot \partial_x L + \frac{\sigma_t(X_t, \pi_t)^2}{2} \partial_{xx} L$$

Substituting in this into the above Bellman equation and letting $\delta \to 0$, 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 77 (Davis-Varaiya Martingale Prinicple 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

$$L_0(X_0) = M_0 \leq \mathbb{E}[M_T] = \underbrace{\mathbb{E}_{X_0}\Big[\int_0^T e^{-\alpha s} c_\tau(X_\tau, \Pi_\tau) d\tau + \underbrace{L_T(X_T)}_{C_T(X_T)}\Big]}_{C(x_0, \Pi)}$$

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

$$L_0(X_0) = \min_{\Pi \in \mathcal{P}} \mathbb{E}\left[\int_0^T e^{-\alpha\tau} c_\tau(X_\tau, \pi_\tau) d\tau + c_T(X_T)\right].$$

□

2.4 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 78 (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

$$dW_t = r \underbrace{(W_t - n_t S_t)}_{\substack{\text{Wealth in} \\ \text{bank}}} dt + \underbrace{n_t dS_t}_{\substack{\text{Wealth in} \\ \text{asset}}} - \underbrace{c_t dt}_{\text{consumption}} \tag{2.6a}$$

$$= r(W_t - n_t \cdot S_t)\, dt + n_t \cdot dS_t - c_t dt \tag{2.6b}$$

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 79 (The Merton Problem – Objective)**.** *Given the above plant equation,* (2.6)*, 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 80. *The HJB equation for the Merton Problem can be written as*

$$0 = \max_c \{u(c) - c\partial_w V\} + \max_\theta \left\{\theta(\mu - r)\partial_w V + \frac{1}{2}\sigma^2\theta^2\partial_{ww}V\right\} - \rho V + rw\partial_w V$$

Here the optimal θ and c are given by

$$\theta^* = -\frac{\partial_w V}{\partial_{ww} V}\sigma^{-2}(\mu - r), \qquad c^* = (u')^{-1}(\partial_w V)$$

Proof. First we note that we can rewrite the SDE for W_t as follows:

$$\begin{aligned} dW_t &= r\left(W_t - n_t \cdot S_t\right)dt + \underbrace{n_t \cdot dS_t}_{=n_t S_t \mu dt + n_t S_t \sigma dB_t} - c_t dt \\ &= \left(rW_t + (\mu - r)\theta_t - c_t\right)dt + \theta_t \sigma dB_t\,. \end{aligned}$$

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

$$\begin{aligned} dV(W_t) &= \partial_w V(W_t)dW_t + \frac{1}{2}\partial_{ww}V(W_t)d[W]_t \\ &= \partial_w V(W_t)\left[r\left(W_t - n_t \cdot S_t\right)dt + n_t \cdot dS_t - c_t dt\right] \\ &+ \partial_t V(W_t)dt + \frac{\theta^2\sigma^2}{2}\partial_{ww}V(W_t)dt \end{aligned}$$

Applying this to the above term gives as required

$$\begin{aligned} 0 &= \max_{\theta,c}\left\{u(c) - \rho V + \left(rw + \theta\cdot(\mu - r) - c\right)\partial_w V + \frac{1}{2}\sigma^2\theta^2\partial_{ww}V\right\} \\ &= \max_c \{u(c) - c\partial_w V\} + \max_\theta\left\{\theta(\mu - r) + \frac{1}{2}\sigma^2\theta^2\partial_{ww}V\right\} - \rho V + rw\partial_w V \end{aligned}$$

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 $\sup_c \{u(c) - c\partial_w V\}$. □

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 81. *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

$$\theta^* = \frac{w}{R}\sigma^{-2}(\mu - r),$$
$$c^* = \gamma^{-\frac{1}{R}}w \quad \textit{and} \quad \sup_c\{u(c) - c\partial_w V\} = \frac{R}{1-R}\gamma^{1-\frac{1}{R}}w^{1-R}.$$

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 80,

$$\theta^* = -\frac{\partial_w V}{\partial_{ww} V}\sigma^{-2}(\mu - r) = \frac{w}{R}\sigma^{-2}(\mu - r)$$

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 80 gives

$$\begin{aligned}0 &= \frac{R}{1-R}\gamma^{1-\frac{1}{R}} w^{1-R} - \frac{1}{2}\sigma^{-2}(\mu - r)^2\frac{(\partial_w V)^2}{\partial_{ww} V} - \rho V + rw\partial_w V \\ &= \frac{R}{1-R}\gamma^{1-\frac{1}{R}} w^{1-R} - \frac{1}{2}\sigma^2(\mu - r)^2\frac{(\gamma w^{-r})^2}{(-R\gamma w^{-R-1})} - \rho\gamma\frac{w^{1-R}}{1-R} \\ &= \gamma w^{1-R}\left[\frac{R}{1-R}\gamma^{\frac{1}{R}} + \frac{1}{2}\sigma^2\frac{(\mu - r)^2}{R} - \frac{\rho}{1-R} + r\right].\end{aligned}$$

Cancelling γw^{1-R} and rearranging gives the required for for γ. □

To summarize: we notice we have shown that the parameters

$$\theta^* = \frac{w}{R}\sigma^{-2}(\mu - r), \quad c^* = \gamma^{-\frac{1}{R}} w, \tag{2.7a}$$

$$\gamma^* = R^{-1}\left\{\rho + (R-1)\left(r + \frac{1}{2}\frac{\kappa^2}{R}\right)\right\}, \quad \kappa = \sigma^{-1}(\mu - r). \tag{2.7b}$$

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 weath under these parameters obeys the SDE

$$dW_t = W_t\left\{R^{-1}\kappa dW_t + (r + R^{-1})|\kappa|^2 - \gamma)dt\right\}$$

which is a geometric Brownian motion:

$$W_t = W_0 \exp\left\{R^{-1}\kappa W_t + (r + \frac{1}{R^2}\kappa^2(2R_1) - \gamma)t\right\}$$

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 82. *The parameters in* (2.7)*, 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

$$\begin{aligned}\mathbb{E}\left[\int_0^\infty e^{-\rho t}u(c_t)dt\right] &\leq \mathbb{E}\left[\int_0^\infty e^{-\rho t}\left\{u(c_t^*) + (c_t - c_t^*)u'(c_t^*)\right\}dt\right] \\ &= \mathbb{E}\left[\int_0^\infty e^{-\rho t}u(c_t^*)dt\right] + \mathbb{E}\left[\int_0^\infty (c_t - c_t^*)\,\zeta_t dt\right] \qquad (2.8)\end{aligned}$$

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

$$\zeta_t = e^{-\rho t}u'(c_t^*) = De^{-\kappa B_t - (r + \frac{\kappa^2}{2})t} \qquad \text{where} \qquad \gamma w_0.$$

Define function

$$f_t(W, B) = W \exp\left\{-\kappa B - \left(r + \frac{\kappa^2}{2}t\right)\right\}$$

and note that $\zeta_t W_t = Df_t(W_t, B_t)$. Now lets apply Ito's formula to $f_t(W_t, B_t)$. By Ito's formula:

$$df = \partial_t f dt + \partial_w f dW_t + \partial_B f dB_t + \frac{1}{2}\partial_B B f d[B]_t + \partial_{Bw} f d[BW]_t + \frac{1}{2}\partial_{ww} f d[W]_t.$$

Now lets check terms.

$$\partial_t f = -\left(r + \frac{1}{2}\kappa^2\right)We^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} \quad \partial_B f = -\kappa We^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} \quad \partial_w f = e^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t}$$

$$\partial_{BB} f = \kappa^2 We^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} \quad \partial_{wB} f = -\kappa e^{-\kappa B_t - (r + \frac{1}{2}\kappa^2)t} \quad \partial_{ww} f = 0$$

$$d[B]_t = dt \quad d[W, B]_t = \theta\sigma dt$$

Substituting these into Ito's formula above gives,

$$df = e^{-\kappa B_t - (r+\frac{1}{2}\kappa^2)t}\Big[- W\Big(r + \frac{1}{2}\kappa^2\Big)dt + \{rW - c_t + \theta(\mu - r)\}\, dt + \theta\sigma dB_t - W\kappa dB_t + \frac{W}{2}\kappa^2 dt - \theta\sigma\kappa dt\Big]$$

Cancelling (using that $\kappa\sigma = (\mu - r)$) and rearrganging we get

$$df + e^{-\kappa B_t - (r+\frac{\kappa^2}{2})t} c_t dt = e^{-\kappa B_t - (r+\frac{1}{2}\kappa^2)t}\,[\theta\sigma - W\kappa]\, dB_t$$

So

$$\zeta_t W_t + \int_0^t \zeta_s c_s ds = Df_t(W_t, B_T) + \int_0^t Dc_t e^{\kappa B_t - (r+\frac{\kappa^2}{2})t} dt$$

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 (2.8) 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. □

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 in invested in. These obey the Stochastic Differential Equation

$$dS_t^i = S_t^i \left\{ \sum_{j=1}^{N} \sigma^{ij} dB_t^j + \mu^i dt \right\}, \quad i = 1, ..., d$$

where B_t^j is an independent Brownian motion for each $j = 1, ..., N$.

Wealth now evolves according the SDE

$$dW_t = r\left(W_t - \boldsymbol{n}_t \cdot \boldsymbol{S}_t\right) dt + \boldsymbol{n}_t \cdot d\boldsymbol{S}_t - c_t dt$$

where $\boldsymbol{n}_t = (n_t^i : i = 1, ..., d)$ gives the amount of each asset $\boldsymbol{S}_t = (S_t^i : i = 1, ..., d)$ held in the portfolio at time t. Also we define $\boldsymbol{\theta}_t = (n_t^i S_t^i : i = 1, ..., d)$ as the wealth in each asset. As given in Def 79, our task is the maximize the objective

$$V(w) = \max_{(\boldsymbol{n}_t, c_t)_{t\geq 0} \in \mathcal{P}(w_0)} \mathbb{E}\left[\int_0^\infty e^{-\rho t} u(c_t) dt \right].$$

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 4. *Show that the HJB equation for the Merton Problem can be written as*

$$0 = \max_c \{u(c) - c\partial_w V\} + \max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \boldsymbol{r}) \partial_w V + \frac{1}{2} |\sigma\boldsymbol{\theta}|^2 \partial_{ww} V \right\} - \rho V + rw\partial_w V.$$

where $\boldsymbol{r} = (r : i = 1, ..., d)$.

Proof. The proof follows more-or-less identically to Prop 80. 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

$$dW_t = \Big(rW_t - \underbrace{r\boldsymbol{n}_t \cdot \boldsymbol{S}_t}_{\boldsymbol{\theta}\cdot\boldsymbol{r}} \Big) dt + \underbrace{\boldsymbol{n}_t \cdot d\boldsymbol{S}_t}_{\boldsymbol{\theta}^\top[\sigma d\boldsymbol{B}_t + \boldsymbol{\mu} dt]} - c_t dt = \left(rW_t + \boldsymbol{\theta}_t(\boldsymbol{\mu} - \boldsymbol{r}) - c_t\right) dt + \boldsymbol{\theta}_t^\top \sigma d\boldsymbol{B}_t$$

$$d[W]_t = \sum_{ij} (\boldsymbol{\theta}_t^\top \sigma)_i (\boldsymbol{\theta}_t^\top \sigma)_j d[B_t^i, B_t^j] = |\boldsymbol{\theta}_t \sigma|^2 dt.$$

Thus

$$dV(W_t) = \left[(rW_t + \boldsymbol{\theta}_t(\boldsymbol{\mu} - \boldsymbol{r}) - c_t)\,\partial_w V(W_t) + \frac{1}{2}|\boldsymbol{\theta}\sigma|^2 \partial_{ww} V(W_t)\right] dt + \boldsymbol{\theta}_t^\top \sigma d\boldsymbol{B}_t.$$

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. □

Lemma 5. *Show the optimal asset portfolio in the HJB equation is given by*

$$\boldsymbol{\theta}^* = -\frac{\partial_w V}{\partial_{ww} V}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \boldsymbol{r})$$

and

$$\max_{\boldsymbol{\theta}} \left\{\boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \boldsymbol{r})\partial_w V + \frac{1}{2}(\boldsymbol{\theta}^\top \sigma^\top \sigma \boldsymbol{\theta})\partial_{ww} V\right\} = -\frac{1}{2}|\boldsymbol{\kappa}|^2 \frac{(\partial_w V)^2}{\partial_{ww} V}$$

Proof. Considering Lemma 4 we have that

$$\max_{\boldsymbol{\theta}} \left\{\boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \boldsymbol{r})\partial_w V + \frac{1}{2}(\boldsymbol{\theta}^\top \sigma^\top \sigma \boldsymbol{\theta})\partial_{ww} V\right\} \implies (\boldsymbol{\mu} - \boldsymbol{r})\partial_w V + \partial_{ww} V(\sigma^\top \sigma)\boldsymbol{\theta}^* = 0.$$

Solving for $\boldsymbol{\theta}^*$ and substituting back into the maximization gives the answer. □

Lemma 6. *Show that for a CRRA utility the optimal solution to the HJB equation is given by*

$$\boldsymbol{\theta}^* = \frac{w}{R}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \boldsymbol{r}), \quad c^* = \gamma^{-\frac{1}{R}} w$$

where

$$\gamma^{-\frac{1}{R}} = R^{-1}\left\{\rho + (R-1)(r + \frac{1}{2}\frac{|\boldsymbol{\kappa}|^2}{R})\right\} \qquad \boldsymbol{\kappa} = \sigma^{-1}(\boldsymbol{\mu} - \boldsymbol{r}).$$

Proof. 6 (In this proof when we refer to Prop 81 we mean that the argument which was applied in the single-asset setting is identical in the multiple asset setting.)

By Prop 81a)

$$V(w) = \gamma \frac{w^{1-R}}{1-R}$$

for some constant γ. Differentiating twice gives

$$\boldsymbol{\theta}^* = -\frac{\partial_w V}{\partial_{ww} V}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \boldsymbol{r}) = \frac{w}{R}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \boldsymbol{r}).$$

By Prop 81b), $c^* = \gamma^{-\frac{1}{R}} w$. Substituting these solutions into the HJB equation gives

$$\begin{aligned} 0 &= \max_c \{u(c) - c\partial_w V\} + \max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \boldsymbol{r})\partial_w V + \frac{1}{2}|\sigma\boldsymbol{\theta}|^2 \partial_{ww} V \right\} - \rho V + rw\partial_w V \\ &= \frac{R}{1-R}\gamma^{1-\frac{1}{R}} w^{1-R} + \frac{1}{2}|\boldsymbol{\kappa}|^2 \cdot \frac{\gamma}{R} w^{1-R} - \rho\gamma \frac{w^{1-R}}{1-R} + r\gamma w^{1-R} \end{aligned}$$

Rearranging and solving for γ gives the required solution for γ^*. □

Def 83 (Merton Portfolio and Market Price Risk Vector)**.** *As given above,*

$$\boldsymbol{\theta}^* = \frac{w}{R}(\sigma\sigma^\top)^{-1}(\boldsymbol{\mu} - \boldsymbol{r}),$$

is called the Merton Portfolio *and*

$$\boldsymbol{\kappa} = \sigma^{-1}(\boldsymbol{\mu} - \boldsymbol{r}).$$

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

$$\lim_{c\to\infty} u'(t, c) = 0$$

The HJB equation for the Merton problem is

$$0 = \max_{\theta, c} \left\{ u(t, c) + \partial_t V + (rw + \theta \cdot (\mu - r) - c)\, \partial_w V + \frac{1}{2}|\sigma^\top \theta|^2 \partial_{ww} V \right\}.$$

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 84. *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

$$\partial_z J = -w, \qquad \partial_{zz} J = -\frac{1}{\partial_{ww} V}, \qquad \partial_t J = \partial_t V \tag{2.9}$$

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 (2.9).

Substituting the expressions in (2.9), the HJB equation is

$$\begin{aligned} 0 &= \max_c \{u(t,z) - c\partial_w V\} + \max_{\boldsymbol{\theta}} \left\{ \boldsymbol{\theta} \cdot (\boldsymbol{\mu} - \boldsymbol{r})\partial_w V + \frac{1}{2}|\sigma\boldsymbol{\theta}|^2 \partial_{ww} V \right\} + \partial_t V + rw\partial_w V \\ &= u^*(t, \partial_w V) - \frac{1}{2}|\boldsymbol{\kappa}|^2 \frac{(\partial_w V)^2}{\partial_{ww} V} + \partial_t V + rw\partial_w V \\ &= u^*(t,z) + \frac{1}{2}|\boldsymbol{\kappa}|^2 z^2 \partial_{zz} J + \partial_t J - rz\partial_z J. \end{aligned}$$

Now is we suppose that $u(t,x) = e^{-\rho t}u(x)$, for $u(x)$ concave and increasing, then by the same argument as Prop 81a) 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:

$$\begin{aligned} &J(t,z) = e^{-\rho t}j(y), && \partial_t J = -\rho e^{-\rho t}j(y) + \rho e^{-\rho t}yj'(y) \\ &\partial_z J = j'(y), && \partial_{zz} J = e^{\rho t}j''(y) \end{aligned}$$

where $y = ze^{\rho t}$. Now substituting these terms into the HJB equation gives the result:

$$0 = \underbrace{u^*(t,z)}_{=e^{-\rho t}u^*(y)} + \underbrace{\partial_t J}_{=-\rho e^{-\rho t}j(y)+\rho e^{-\rho t}yj'(y)} - \underbrace{rz\partial_z J}_{re^{-\rho t}yj'(y)} + \underbrace{\frac{1}{2}|\kappa|^2 z^2 \partial_{zz} J}_{\frac{1}{2}|\kappa|^2 y^2 e^{-\rho t}j''(y)}\,.$$

□

Exercises

Ex 85. *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 86. *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 87. *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:

$$V(w,r) = \max_{(\theta_s,c_s)_{s\geq 0}} \mathbb{E}\left[\int_0^\infty e^{-\rho s}u(c_s)ds \middle| W_0 = w, r_0 = r\right], \quad \text{with} \quad u(c_s) = \frac{c_s^{1-R}}{1-R}.$$

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 3

Stochastic Approximation

3.1 Robbins-Munro.

- Robbins Munro 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 Munro consider the setting where we cannot directly observe g but we might observe some random variable whose mean is $g(x)$. Thus we observe

$$y_n = g(x_n) + \epsilon_n \tag{3.1}$$

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 Munro 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

$$\sum_n \alpha_n = \infty, \qquad \sum_n \alpha_n^2 < \infty\,.$$

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 preform 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-Munro proof

Lem 88. *Suppose that z_n is a positive sequence such that*

$$z_{n+1} \leq z_n(1 - a_n) + c_n \tag{3.2}$$

where a_n and c_n are positive sequences such that

$$\sum_n a_n = \infty, \qquad \text{and} \qquad \sum_n c_n < \infty \tag{3.3}$$

then $\lim_{n\to\infty} z_n = 0$.

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 (3.2); neither of which effect the conditions on a_n and b_n, (3.3).

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 $\lim_n z_n = 0$. □

An Easy Robbin's Munro Proof

The following lemma is a straight-forward proof based on the original argument of Robbins and Munro. 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 $\kappa > 0$

$$(g(y) - g(x))^\top (y - x) \geq \kappa \|y - x\|^2 \tag{3.4}$$

Here are a couple of instances where this holds:

- $g(x)$ is the gradient of a strongly convex function $f(x)$.[1]
- $g : [x_{\min}, x_{\max}] \to [y_{\min}, y_{\max}]$ 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-Munro 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 89 (Robbins-Munro)**.** *If we chose x_n acording to the Robbin-Munro step rule* (RM) *and we assume* (3.4) *then we have that*

$$\mathbb{E}[(x_n - x^*)^2] \xrightarrow[n\to\infty]{} 0$$

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

$$z_{n+1} \leq z_n(1 - 2a_n\kappa) + a_n^2 e_n.$$

Now applying Lemma 88 gives the result. □

Additional Lemmas

The following Lemma is also sometimes applied to get tighter bounds on convergence.

[1] Note, if $g(x) = \nabla f(x)$ then (3.4) is the definition of $f(x)$ begin strongly convex.

Lem 90. *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

$$\begin{aligned} z_{n+1} \leq z_n(1 - a_n) + c_n &\leq (z_{n-1}(1 - a_{n-1}) + c_{n-1})(1 - a_n) + c_n \\ &\leq z_{n-1}(1 - a_{n-1})(1 - a_n) + c_n + c_{n-1}(1 - a_n) \\ &\vdots \\ &\leq z_0 \prod_{k=0}^{n}(1 - a_k) + \sum_{j=0}^{n} c_j \prod_{k=j+1}^{n}(1 - a_k) \end{aligned}$$

□

The following proposition is a Martingale version of the above result.

Prop 91 (Robbins-Siegmund Theorem)**.** *If*

$$\mathbb{E}[Z_{n+1}|\mathcal{F}_n] \leq (1 - a_n + b_n)Z_n + c_n \tag{3.5}$$

for positive adaptive RVs Z_n, a_n, b_n, c_n such that with probability 1,

$$\sum_n a_n = \infty, \qquad \sum_n b_n < \infty, \qquad \textit{and} \qquad \sum_n c_n < \infty$$

then $\lim_{n\to\infty} z_n = 0$.

Proof. The results is some manipulations analogous to the Robbins-Munro 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 (3.5) 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 Convergnce Theorem, $\lim_{n\to\infty} Y_{n\wedge\tau_C}$ exists for each $C > 0$, and $\tau_C = \infty$ for some C, since $\sum c'_k < \infty$. Thus $\lim_{n\to\infty} Y_n$ 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. □

Stochastic Approximation Examples.

Ex 92. *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

$$z_t - z_1 \leq e^{-\int_0^t \alpha_s ds} + \int_0^t \frac{1}{2}\alpha_s^2 e^{-\int_s^t \alpha_u du} ds.$$

We analyzing the 2nd term in this expression. Suppose that $\alpha_s = \frac{1}{s^\gamma}$*, using Integration by parts or otherwise, show that*

$$\int_1^t \frac{1}{2}\alpha_s^2 e^{-\int_s^t \alpha_u du} ds \leq \frac{1}{t^\alpha} + \frac{2}{t} - 2e^{-\int_1^t s^{-\gamma} ds}\,.$$

Rmk 93. *A quick remark before proceeding with the solution. Note that the above SDE behaves very similarly to the Robbins-Munro 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 [33] (A very readable paper). Stochastic approximation has grown enormously, see Krushner and Yin [24] for an excellent text on the topic. The discussion on finite time error is based on Bach and Moulines [1]. Asynchronous update section is based on reading Tsitsiklis [40] (and Bertsekas & Tsitsiklis [6]), here we apply a Robbins & Siegmund [34].

3.2 Stochastic Gradient Decent

Suppose that we have some function $F : \mathbb{R}^p \to \mathbb{R}$

$$F(\theta) = \mathbb{E}_X[f(X;\theta)]$$

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 $X_1, X_2, \ldots$. 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

$$\sum_n \alpha_n = \infty, \qquad \sum_n \alpha_n^2 < \infty\,.$$

(Note here we may assume that α_n is a function of previous parameters and observations $\theta_1, ..., \theta_{n-1}$ and $X_1, ..., X_{n-1}$.) We let $\|\cdot\|_2$ be the Euclidean norm. We can prove that convergence θ_n to the minimizer of $F(\theta)$.

Thrm 94 (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 expecations 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 91, $\theta_{n+1} \to \theta^*$. 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 88 (note that $a_n = \alpha_n\mu + \alpha_n^2 B$ will be positive for suitably large n), to give that $\mathbb{E}\|\theta_{n+1} - \theta^*\|_2^2] \to 0$ as $n \to \infty$ as required. □

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

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \leq (1 - \alpha_n\mu + \alpha_n^2 B)\mathbb{E}[\|\theta_n - \theta^*\|_2^2] + \alpha_n^2 C \tag{3.6}$$

that we found in our proof of Theorem 94. (Here $C = K + A$.)

Thrm 95. *There exist positive constants* a_1, a_2, M_1 *and* M_2

$$\mathbb{E}[\|\theta_{n+1} - \theta^*\|_2^2] \leq \frac{4C}{\mu}\frac{1}{n^\gamma} + \mathbb{E}[\|\theta_0 - \theta^*\|_2^2]M_1 e^{-a_1 n^{1-\gamma}} + n^{1-2\alpha}M_2 e^{-a_2 n^{1-\gamma}}. \tag{3.7}$$

Proof. Letting $z_n = \mathbb{E}[\|\theta_n - \theta^*\|_2^2]$. Applying Lemma 88 gives that

$$z_{n+1} \leq z_0 \underbrace{\prod_{k=0}^{n}(1 - \alpha_k\mu + \alpha_k^2 B)}_{=:F_n} + \underbrace{\sum_{m=0}^{n} C\alpha_m^2 \prod_{j=m+1}^{n}(1 - \alpha_k\mu + \alpha_k^2 B)}_{=:G_n}$$

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:

$$\begin{aligned}
G_n &= \sum_{m=n/2}^{n} C\alpha_m^2 \prod_{j=m+1}^{n}(1 - \alpha_k\mu + \alpha_k^2 B) + \sum_{m=0}^{n/2} C\alpha_m^2 \prod_{j=m+1}^{n}(1 - \alpha_k\mu + \alpha_k^2 B) \\
&\leq \frac{2C}{\mu}\alpha_{n/2} \sum_{m=n/2}^{n} \underbrace{\frac{\alpha_k\mu}{2}\prod_{j=m+1}^{n}\left(1 - \frac{\alpha_k\mu}{2}\right)}_{=\prod_{j=m+1}^{n}(1-\frac{\alpha_k\mu}{2}) - \prod_{j=m}^{n}(1-\frac{\alpha_k\mu}{2})} + \left(\sum_{m=0}^{n} C\alpha_m^2\right)\prod_{j=n/2}^{n}(1 - \frac{\alpha_k\mu}{2}) \\
&\leq \frac{2C}{\mu}\alpha_{n/2}\left[1 - \prod_{j=n/2}^{n}\left(1 - \frac{\alpha_k\mu}{2}\right)\right] + \left(\sum_{m=0}^{n} C\alpha_m^2\right)\exp\left\{-\frac{\mu}{2}\sum_{k=n/2}^{n}\alpha_k\right\}.
\end{aligned}$$

Putting our bounds on F_n and G_n together then gives

$$z_{n+1} \leq \frac{2C}{\mu}\alpha_{n/2} + z_0 \exp\left\{-\frac{\mu}{2}\sum_k \alpha_k\right\} + \left(\sum_{m=0}^{n} C\alpha_m^2\right)\exp\left\{-\frac{\mu}{2}\sum_{j=n/2}^{n}\alpha_k\right\}$$

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

$$z_{n+1} \leq \frac{4C}{\mu n^{\gamma}} + z_0 M_1 \exp\left\{-\frac{\mu n^{1-\gamma}}{2(1-\gamma)}\right\} + n^{1-2\alpha}M_2 \exp\left\{-\frac{\mu(1-\left(\frac{1}{2}\right)^{1-\gamma})}{2(1-\gamma)}n^{1-\gamma}\right\}.$$

□

Rmk 96. *Notice the order of magnitude achieved is correct (assuming the original inequality in* (3.6) *is tight). To see this notice that the product for G_n above behaves as the intergral to which we can apply integration by parts:*

$$\begin{aligned}\int_1^n \frac{1}{x^{2\alpha}} e^{-\int_x^n \frac{1}{y^\alpha}dy}dx &= \int_1^n \underbrace{\frac{1}{x^\alpha}}_{u}\underbrace{\frac{1}{x^\alpha}e^{-\int_x^n \frac{1}{y^\alpha}dy}}_{dv}dx \\ &= \underbrace{\left[\frac{1}{x^\alpha}e^{-\int_x^n \frac{1}{y^\alpha}dy}\right]_1^n}_{\frac{1}{n^\alpha}-e^{-\int_1^n y^\alpha dy}} + \int_1^n \frac{\alpha}{x^{\alpha-1}}e^{-\int_x^n \frac{1}{y^\alpha}dy} \geq \frac{1}{n^\alpha}\end{aligned}$$

which suggests that the convergence rate of $\frac{1}{n^\alpha}$ found for this term is of the correct order of magnitude.

3.3 Lyapunov Functions

Lyapunov functions are an extremely convenient device for proving that a dynamical system converges.

- For some continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$, we suppose $x(t)$ obeys the differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t)), \qquad t \in \mathbb{R}_+. \tag{3.8}$$

- A Lyapunov function is a continuously differentiable function $L : \mathbb{R}^n \to \mathbb{R}$ with unique minimum at x^* such that

$$f(x) \cdot \nabla L(x) < 0, \qquad \forall x \neq x^*. \tag{3.9}$$

- We add the additional assumption that $\{x : L(x) \leq l\}$ is a compact set for every $l \in \mathbb{R}$.

Theorem 1. *If a Lyapunov exists* (3.9) *for differential equation* (3.8) *then* $L(x(t)) \searrow L(x^*)$ *as* $t \to \infty$ *and*

$$x(t) \xrightarrow[t\to\infty]{} x^*.$$

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

$$\tilde{L} - L(x(t)) = \int_t^\infty \frac{dL(x(s))}{ds} ds \xrightarrow[t\to\infty]{} 0 \tag{3.10}$$

Thus we can take a sequence of times $\{s_k\}_{k=1}^\infty$ such that $\frac{dL(x(s_k))}{dt} \to 0$ as $s_k \to \infty$. As $\{x : L(x) < L(x(0))\}$ is compact, we can take a subsequence of times $\{t_k\}_{k=1}^\infty \subset \{s_k\}_{k=1}^\infty$, $t_k \to \infty$ such that $\{x(t_k)\}_{k=0}^\infty$ converges. Suppose it converges to $\tilde{x}$. By continuity,

$$0 = \lim_{k\to\infty} \frac{dL(x(t_k))}{dt} = \lim_{k\to\infty} f(x(t_k)) \cdot \nabla L(\tilde{x}) = f(\tilde{x}) \cdot \nabla L(\tilde{x}).$$

Thus by definition $\tilde{x} = x^*$. Thus $\lim_{t\to\infty} L(x(t)) = L(x^*)$ and thus by continuity of L at x^* we must have $x(t) \to x^*$. □

- One can check this proof follows more-or-less unchanged if x^*, the minimum of L, is not unique.

La Salle's Invariance Prinicple

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

$$\mathcal{X}^\star = \{x : g(x) \cdot \nabla L(x) = 0\}$$

The set $\mathcal{X}^\star$ is called the set of *invariant points*. Notice, under the conditions of Theorem 1, $\mathcal{X}^\star = \{x^\star\}$. In general, $\mathcal{X}^\star$ 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 $\mathcal{X}^\star$ is exactly the set of local minima. The result which proves convergence to invariant points is called *Salle's Invariance Prinicple*.

We assume the following

- The process $x(t)$, $t \in \mathbb{R}_+$ obeys the o.d.e.

  $$\frac{dx}{dt} = g(x(t))$$

  for $g : \mathbb{R}^d \to \mathbb{R}^d$ 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 $L : \mathbb{R}^d \to \mathbb{R}_+$ is a continuously differentiable function such that

  $$g(x) \cdot \nabla L(x) \leq 0, \qquad \forall x \in \mathcal{X}$$

  and we let

  $$\mathcal{X}^\star = \{x \in \mathcal{X} : g(x) \cdot \nabla L(x) = 0\}.$$

La Salle's Invariance Prinicple is as follows[2]

Theorem 2 (La Salle's Invariance Prinicple)**.** *As $t \to \infty$, $x(t)$ convergences to $\mathcal{X}^\star$ 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*

$$\min_{x^\star \in \mathcal{X}^\star} \left| x(T) - x^\star \right| < \epsilon \,.$$

[2]The orginial 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 $\mathcal{X}^\star(\delta) = \{x \in \mathcal{X} : -g(x) \cdot \nabla L(x) < \delta\}$. Since $dL/dt = g(x) \cdot \nabla L(x)$ and $L(x_0) \leq l$, for time $T > L/\delta$ is must be that $x(t) \in \mathcal{X}^\star(\delta)$ for all $t \geq T$. [This is the main part of the argument completed.]

It is reasonable to assume that $x(t) \in \mathcal{X}^\star(\delta)$ for δ suitably small then $x(t)$ must be close to $\mathcal{X}^\star$. This is true and to show this we prove the following claim: for $\epsilon > 0$ $\exists \delta > 0$ such that if $x(t) \in \mathcal{X}^\star(\delta)$ then $|x(T) - x^\star| < \epsilon$ for some $x^\star \in \mathcal{X}^\star$. 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 $|x_n - x^\star| \geq \epsilon$ for all $x^\star \in \mathcal{X}^\star$. Since $\mathcal{X}$ is compact $x_{n_k} \to x_\infty$ over some subsequence $\{n_k\}_{k \in \mathbb{N}}$. By continuity $g(x_\infty) \cdot \nabla L(x_\infty) = 0$ and $|x_\infty - x^\star| \geq \epsilon$ for all $x^\star \in \mathcal{X}$, which is a contradiction since $g(x_\infty) \cdot \nabla L(x_\infty) = 0$ means $x_\infty \in \mathcal{X}^\star$. This contradiction thus such thata there there exists a value $n^\star$ such that for all x such that $-g(x) \cdot \nabla L(x) < \delta := 1/n^\star$ implies $\min_{x^\star} |x - x^\star| < \epsilon$. □

Remark 97. *We assume $x(t) \in \mathcal{X}$ for some compact set $\mathcal{X}$. Notice a natural condition that implies compactness is that*

$$\liminf_{|x| \to \infty} L(x) = \infty .$$

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 $x(t) \in X_l$ 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 3. *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 $L(x(t))$.

$$\int_0^T L(x(t))dt = \int_0^T L(x(t)) - L(x^\star)dt \leq \int_0^T \nabla L(x(t))(x(t)-x^\star)dt$$

Notice that since $\frac{dx}{dt} = -\eta_t \nabla L(x(t))$ we have

$$-\frac{1}{2\eta_t}\frac{d}{dt}\|x(t)-x^\star\|^2 = \nabla L(x(t))(x(t)-x^\star).$$

Substituting this into the integral above gives

$$\begin{aligned}
\int_0^T L(x(t))dt &\leq -\int_0^T \frac{1}{2\eta_t}\frac{d}{dt}\|x(t)-x^\star\|^2 dt \\
&= -\left[\frac{1}{2\eta_t}\|x(t)-x^\star\|^2\right]_0^T + \int_0^T \frac{1}{2}\|x(t)-x^\star\|^2 \frac{d\eta^{-1}}{dt}dt \\
&\leq \frac{D}{2\eta_T} + \frac{D}{2\eta_T} = \frac{D}{\eta_T}
\end{aligned}$$

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 $F : \mathcal{R}^d \to \mathbb{R}^d$ 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 4. *If we suppose that $x(t)$ obeys the ODE*

$$\frac{dx}{dt} = F(x) - x$$

where $F(x)$ is contraction with fixed point $x^\star$ then $L(x) = \|x - x\|_p$ is a Lyapunov function and $x(t) \to x^*$.

Proof. We let $sgn(x)$ be the sign function that is $sgn(x) = +1$ if $x > 0$, $sgn(x) = -1$ if $x < 0$ and $sgn(x) = 0$ if $x = 0$. For now we assume $1 \leq p < \infty$ (shortly we will extend to allow $p = \infty$).

By the chain rule

$$\begin{aligned}
\frac{dL}{dt} &= \sum_i \frac{\partial L}{\partial x_i} \frac{dx_i}{dt} \\
&= \underbrace{\left(\sum_j w_j |x_j - x^\star|^p \right)^{\frac{1-p}{p}}}_{\|x - x^\star\|_p^{1-p}} \sum_i w_i sgn(x_i - x_i^\star) |x_i - x^\star|^{1-p} \underbrace{\Big(F_i(x) - x_i\Big)}_{F_i(x) - F_i(x^\star) - (x_i - x_i^\star)} \\
&\|x - x^\star\|_p^{1-p} \underbrace{\left[\sum_i w_i sgn(x_i - x_i^\star) |x_i - x_i^\star|^{p-1} \Big(F_i(x) - F_i(x^\star)\Big) \right]}_{\leq \|x - x^\star\|^{p-1} \|F(x) - F(x^\star)\|_p} - \|x - x^\star\|_p \\
&\leq \|F(x) - F(x^\star)\|_p - \|x - x^\star\|_p
\end{aligned}$$

Therefore integrating the above gives

$$||x(t) - x^\star||_p - ||x(s) - x^\star||_p \leq \int_s^t ||F(x(u)) - F(x^\star)||_p - ||x(u) - x^\star||_p du$$

Notice we can allow $p = \infty$ in the above expression since $||z||_p \to ||z||_\infty$ uniformly on compacts. So applying the fact F is a contraction we gain that

$$||x(t) - x^\star||_p - ||x(s) - x^\star||_p \leq -(1 - \beta) \int_s^t ||x(u) - x^\star||_p du$$

which ensures convergence of $x(t)$ to $x^\star$. □

Exponential Convergence

We now place some assumptions where we can make further comments about rates of convergence.

Theorem 5. *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* $t \in \mathbb{R}_+$

$$\|x(t) - x^*\| \leq Ke^{-\kappa t}. \tag{3.11}$$

Proof.

$$\frac{\mathrm{d}L(x(t))}{\mathrm{d}t} = f(x(t)) \cdot \nabla L(x(t)) \leq -\gamma L(x(t)).$$

So long as $x(t) \neq x^*$, $L(x(t)) > 0$, thus dividing by $L(x(t))$ and integrating gives

$$\log L(x(t)) - \log L(x(0)) = \int_0^t \frac{1}{L(x(s))} \frac{\mathrm{d}L(x(s))}{\mathrm{d}t} ds \leq -\gamma t$$

Rearrganging 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 arround x^*. We have convergence from Theorem 1, 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^*.

References

The Lyapunov method goes back to Lyapunov in (1892) [29]. Extensions were considered by La Salle [26]. A widely used textbook treatment is Khalil [23]. Applications to internet congestion control are given by Srikant [36]. The convex Lyapunov function proof is the an o.d.e adaptation to the result of [48].

3.4 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

$$\sum_{n=0}^{\infty} \alpha_n = \infty, \qquad \sum_{n=0}^{\infty} \alpha_n^2 < \infty\,. \tag{RM}$$

Put informally, the condition $\sum_n \alpha_n$ is used to keep the process moving [albeit inn 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 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

$$x(t) = x_n, \qquad \text{for } t = t_n.^3$$

[3] We 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

$$\sup_{t\in[t_m,t_m+T]\cap\mathcal{T}} \|x(t) - z_m(t)\|.$$

Assumptions. In addition to the Robbins-Monro condition (RM), we make the following assumptions.

- g is Lipschitz continuous.
- For $F_n = (x_m, \epsilon_{m-1} : m \le n)$

  $$\mathbb{E}[\epsilon_n | F_n] = 0$$

  and $\sup_n \mathbb{E}[\epsilon^2] < \infty$.

Main result. A key result that we will prove is the following proposition

Proposition 1.

$$\lim_{m\to\infty} \sup_{t\in[t_m,t_m+T]\cap\mathcal{T}} \|x(t) - z_m(t)\| = 0.$$

where convergence holds with probability 1 *and in* L^2.[4]

Proof. Notice

$$z_m(t_n) = x_m + \int_{t_m}^{t_n} z_m(u)du$$

while

$$x(t_n) = x_m + \sum_{k=m}^{n-1} \alpha_k g(x_k) + \sum_{k=m}^{n-1} \alpha_k \epsilon_k = x_m + \int_{t_m}^{t_n} g(x(\lfloor u \rfloor_\alpha))du + \sum_{k=m}^{n-1} \alpha_k \epsilon_k$$

[4]Convergence in L^2 occur assuming $\|\cdot\|$ is the usual Euclidean distance.

where $\lfloor u \rfloor_\alpha = \max\{t_n : t_n \le u\}$. So

$$\begin{aligned}\|z_m(t_n) - x(t_n)\| &\le \Big\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \Big\| + \int_{t_m}^{t_n} \|g(z_m(u)) - g(x(\lfloor u \rfloor_\alpha))\| du \\ &\le \Big\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \Big\| + \int_{t_m}^{t_n} \|z_m(u) - x(\lfloor u \rfloor_\alpha)\| du\end{aligned}$$

which implies by Gronwall's Lemma [Theorem 123]

$$\|z_m(t_n) - x(t_n)\| \le \Big\| \sum_{k=m}^{n-1} \alpha_k \epsilon_k \Big\| e^{L(t_n - t_m)} \le \Big\| M_n - M_m \Big\| e^{LT} \,. \tag{3.12}$$

where the final inequality holds for t_n such that $t_n - t_m \le T$, 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

$$\mathbb{E} M_n^2 = \sum_{k=1}^n \alpha \mathbb{E}[\epsilon_k^2] \le K \sum_{k=1}^\infty \alpha_k^2 < \infty$$

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

$$\lim_{m \to \infty} \sup_{n \ge m} \|M_n - M_m\| = 0$$

with probability 1 and in L^2. Applying this to (3.12) gives the required result

$$\lim_{m \to \infty} \sup_{t \in [t_m, t_m + T] \cap \mathcal{T}} \|z_m(t) - x(t)\| = 0 \,.$$

□

Applying the o.d.e limit. Before we focus on the proof of Proposition 1, 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 $\mathcal{X}^\star$ in T time units for each initial condition x_n, $n \in \mathbb{N}$.

2. Then apply Proposition 1 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 $\mathcal{X}^\star$.

Here, we combine La Salle's Invariance Principle, Theorem 2, with Proposition 1, though, in principle, we could have considered any of the ode results from Section 3.3. We assume that the assumptions of Proposition 1 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 $L : \mathbb{R}^d \to \mathbb{R}_+$ 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 $\mathcal{X}^\star := \{x : g(x) \cdot \nabla L(x) = 0\}$. 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 $\max_{x^\star \in \mathcal{X}^\star} |z(t) - x^\star| \leq \epsilon$ for all $t \geq T$.

Theorem 6. *For the stochastic approximation scheme*

$$x_{n+1} = x_n + \alpha_n[g(x_n) + \epsilon_n]$$

described in Proposition 1 and given a Lyapunov function L as described above, it holds, with probability 1,

$$x_n \xrightarrow[n\to\infty]{} \mathcal{X}^\star$$

Proof. First we check that the o.d.e solutions $z_m(t)$ considered in Proposition 1 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 $l = \max\{L(x) : \|x\| \leq X\}$ 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 $\forall m$

$$\min_{x^\star} |z_m(t_m + t) - x^\star| \leq \epsilon\,.$$

Taking this choice of T, we know from Proposition 1 that there exists $m^\star$ s.t. $\forall m > m^\star$

$$\sup_{t\in[t_m,t_m+2T]\cap\mathcal{T}} \|x(t) - z_m(t)\| \leq \epsilon.$$

Notice we can take $m^\star$ suitably large so that $\alpha_m =: t_m - t_{m-1} \leq T$ for all $m \geq m^\star$. Notice this implies that

$$\bigcup_{m:m\geq m^\star} [t_m + T, t_m + 2T] = [t_{m^\star} + T, \infty)\,.$$

Now notice that if n is such that $t_n \in [t_m + T, t_m + 2T]$ for some $m \geq m^\star$ then combining the inequalities above, gives that

$$\min_{x^\star} \|x_n - x^\star\| \leq \|x_n - z_m(t_n)\| + \min_{x^\star} \|z_m(t_n) - x^\star\| \leq 2\epsilon\,.$$

Thus we see, with the union above, that for all $t_n \geq t_m + T$ it holds that $\min_{x^\star} \|x_n - x^\star\| \leq 2\epsilon$. In other words $x_n \to \mathcal{X}^\star$ as required. □

References

The o.d.e approach to stochastic approximation was initiated by Ljung [28]. Shortly after it is was extensively developed by Kushner, see [25] and [24] for two text book accounts. The arguments above loosely follow the excellent text of Borkar [9]. We shorten the proof in several ways and consider L^2 convergence. A further text with a general treatment is Benveniste et al. [4].

3.5 Asynchronous Update

We now consider Robbins-Munro from a slightly different perspective. Suppose we have a continuous function $F : \mathbb{R}^p \to \mathbb{R}^p$ 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 $\beta \in (0,1)$,

$$\|F(x) - F(y)\|_\infty \leq \beta \|x - y\|_\infty \,. \tag{3.13}$$

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 $i = 1, \dots p$:

$$x_i(t+1) = x_i(t) + \alpha_i(t)(F_i(x(t)) - x_i(t) + \epsilon_i(t)) \tag{RM-Async}$$

where $\alpha_i(t)$ is a sequence such that for all i

$$\sum_t \alpha_i(t) = \infty, \qquad \sum_t \alpha_i^2(t) < \infty \,. \tag{RM step}$$

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

$$\mathbb{E}[\epsilon_i(t) | \mathcal{F}_t] = 0 \,. \tag{3.14}$$

We assume both the functions $F(x)$ and the noise $\epsilon_i(t)$ are bounded.[5]

Asynchonous 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 7. *Suppose that $F(\cdot)$ is a contraction with respect to $\|\cdot\|_\infty$* (3.13)*, suppose the vector $x(t)$ obeys the step rule* (RM-Async) *with*

[5] However, we remark that this assumption can be relaxed significantly. At some expense in doubling the length of the proof. See [40].

step sizes satisfying (RM step) *and further suppose that* $F(x)$ *and noise terms are bounded, then*

$$\lim_{t\to\infty} x(t) = x^*$$

where x^* *is the fixed point* $F(x^*) = x^*$.

Proof of Theorem 7. 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 $\|x(t)\|_\infty \le D_0$ 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 $D_k \to 0$. By induction, we will show that, given $\|x(t)\|_\infty < D_k$ for all $t \ge \tau_k$ for some τ_l, then there exists a τ_{k+1} such that for all $t \ge \tau_{k+1}$

$$\|x(t)\|_\infty < D_{k+1}$$

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 $t \ge \tau_k$, 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

$$|x_i(t) - W_i(t)| \le Y_i(t)$$

in Lemma 7 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 7. $\forall t_0 \ge \tau_k$

$$|x_i(t) - W_i(t)| \le Y_i(t)$$

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. □

Lemma 8.

$$\lim_{t\to\infty} |W_i(t)| = 0$$

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 91), we know that

$$\lim_{t\to\infty} W(t) = 0.$$

□

Lemma 9.

$$Y_i(t) \xrightarrow[t\to\infty]{} \beta D_k$$

Proof. Notice

$$Y_i(t+1) - \beta D_k = (1-\alpha_i(t))(Y_i(t) - \beta D_k) = ... = \left(\prod_{s=1}^{t}(1-\alpha_i(s))\right)(Y_i(0) - \beta D_k)$$

The result holds since $\sum_t \alpha_i(t) = \infty$. □

We can now prove Theorem 7.

Proof of Theorem 7. 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 7 and then by Lemmas 8 and 9

$$|x_i(t)| \leq Y_i(t) + |W_i(t)| \xrightarrow[t\to\infty]{} \beta D_k$$

Thus these 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 $k \to \infty$, thus $x(t)$ goes to zero as required. □

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:

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

or evaluating the reward function of the current policy

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

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 incase 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^a_{xy} for each action. For instance, if we perform policy improvement based on an estimation of the value function to V,

$$\pi(x) \in \operatorname*{argmax}_{a \in \mathcal{A}} \; r(x,a) + \sum_{\hat{x}} P^a_{x\hat{x}} V(\hat{x})$$

then this is not model free, because we need to estimate $P^a_{x\hat{x}}$ 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 98 (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 99 (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, the 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 100 (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

$$R(x) \leftarrow R(x) + \sum_{v=1}^{N} \alpha d(\hat{\boldsymbol{x}}^{(v)})$$

i.e. we only update R once at the end of the episode, but the update applies a term for every visit $v = 1, ..., N$ to x. While in an online every-visit algorithm, we update

$$R(x) \leftarrow R(x) + \alpha d(\boldsymbol{x})$$

When we talk above different policy evaluation algorithms we can talk about offline & online and first-visit & every-visit variants. From a theoretical perspective offline first-visit algocrithms 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 [39] . Though to go a little deeper, I have benefited a lot from reading the more mathematically rigorous text of Bertsekas and Tsitsiklis [6]. 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 1.5 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

$$R(x) \leftarrow R(x) + \alpha d(\hat{\boldsymbol{x}}) \tag{4.1}$$

for each state x, where $\hat{\boldsymbol{x}} = (\hat{x}_0, \hat{x}_1, ...)$ 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 10 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 101 (Monte-Carlo Policy Evaluation)**.** *When x is visited in an episode update*

$$N(x) \leftarrow N(x) + 1$$
$$\bar{R}(x) \leftarrow \bar{R}(x) + \frac{1}{N(x)}\left(\tilde{R}(x) - \bar{R}(x)\right)$$

where

$$\tilde{R}(x) = r(\hat{x}_0) + \beta r(\hat{x}_1) + ...\beta^T r(\hat{x}_T)$$

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 2.

$$\bar{R}(x) \xrightarrow[N(x)\to\infty]{} R(x) = \mathbb{E}[\tilde{R}(x)] .$$

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.
[1]

[1]It 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 102 (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:

$$\bar{R}(x) = \alpha\tilde{R}_N + \alpha(1-\alpha)\tilde{R}_{N-1} + \alpha(1-\alpha)^2\tilde{R}_{N-2} + ... + \alpha(1-\alpha)^{N-1}\tilde{R}_1.$$

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 10. *For data $a_1, a_2, ...$, 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

$$\begin{aligned} N \times \bar{a}_N &= N \times \left(\bar{a}_{N-1} + \frac{1}{N}\left(a_N - \bar{a}_{N-1}\right)\right) \\ &= a_N + (N-1)\bar{a}_{N-1} \\ &= ... = \sum_{n=1}^{N} \tilde{a}_n. \end{aligned}$$

Above we can repeat the same substitution on $(N-1)a_{N-1}$ as we did for Na_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

$$R(x) = \mathbb{E}_x[r(x, \hat{x}) + \beta R(\hat{x})] \tag{4.2}$$

or, equivalently,

$$R(x) = R(x) + \mathbb{E}\left[r(x, \hat{x}) + \beta R(\hat{x}) - R(x)\right]$$

which is a fixed point of the operation

$$R(x) \overset{\alpha}{\leftarrow} \mathbb{E}\left[r(x, \hat{x}) + \beta R(\hat{x}) - R(x)\right]$$

which, in turn, can be approximated by

$$R(x) \overset{\alpha}{\leftarrow} r(x, \hat{x}) + \beta R(\hat{x}) - R(x).$$

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 7 the algorithm converges to the correct reward function. We summarize each of these points below.

Def 103 (Temporal Differences)**.** *The above term*

$$d(x, x') = r(x, x') + \beta R(x') - R(x)$$

is called a temporal difference.

Def 104 (TD(0))**.** *The algorithm where on every visit to x we perform the update:*

$$R(x) \overset{\alpha}{\leftarrow} r(x, \hat{x}) + \beta R(\hat{x}) - R(x)$$

is called TD(0). Here TD stands for Temporal Difference.

Theorem 8. *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

$$R_t(x) \xrightarrow[t\to\infty]{} R(x) := \mathbb{E}_x\left[\sum_{s=0}^{\infty} \beta^s r(x_s, x_{s+1})\right]$$

where the reward function, satisfying (4.2)*.*

Proof. Notice that $R_t(x)$ follows the update

$$R_{t+1}(x_t) = \alpha_t(x_t)\left[r(x_t, x_{t+1}) + \beta R(x_{t+1})\right]$$

and $R_{t+1}(x') = R_t(x')$ for all $x' \neq x_t$. This is an asynchronous Robbins-Munro scheme, and by Theorem 7 convergence almost surely to $R(x)$ satifying 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 13 (see the proposition subsequent remark too), is equal to $\mathbb{E}_x[\sum_{s=0}^{\infty} \beta^s r(x_s, x_{s+1})]$. □

Remark 105. *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 [40, 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:

$$
\begin{aligned}
R(\hat{x}_0) &= R(\hat{x}_0) + \mathbb{E}\left[r(\hat{x}_0, \hat{x}_1) + \beta R(\hat{x}_1) - R(\hat{x}_0)\right] \\
&\vdots \\
&= R(\hat{x}_0) + \mathbb{E}\left[\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t, \hat{x}_{t+1}) + \beta^n R(\hat{x}_n) - R(\hat{x}_0)\right] \\
&= R(\hat{x}_0) + \mathbb{E}\left[\sum_{t=0}^{n-1} \beta^t d(\hat{x}_t, \hat{x}_{t+1})\right]
\end{aligned}
$$

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 106 (n-Step TD)**.** *The update*

$$
R(x) \overset{\alpha}{\leftarrow} \sum_{t=0}^{n-1} \beta^t d(\hat{x}_t, \hat{x}_{t+1}) = \sum_{t=0}^{n-1} \beta^t r(\hat{x}_t, \hat{x}_{t+1}) + \beta^n R(\hat{x}_n) - R(\hat{x}_0)
$$

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 11 (Bias-Variance Decomposition for n-step TD)**.**

$$
\mathbb{E}[TD_n^2] = \underbrace{\beta^{2n} \mathbb{E}_x[R^\pi(\hat{x}_n) - R(\hat{x}_n)]^2}_{\textit{Bias}} + \underbrace{\beta^{2n} \mathbb{V}(R(\hat{x}_n)) + \mathbb{V}\Big(\textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\Big)}_{\textit{Variance}}
$$

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.

$$\begin{aligned}
\mathbb{E}_x[TD_n^2] &= \mathbb{E}_x\left[\left(R^\pi(\hat{x}_0) - \beta^n R(\hat{x}_n) - \textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right)^2\right] \\
&= \mathbb{E}_x\left[R^\pi(\hat{x}_0) - \beta^n R(\hat{x}_n) - \textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right]^2 \\
&\quad + \mathbb{E}_x\left[\left(\beta^n R(\hat{x}_n) + \textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) - \mathbb{E}_x\left[\beta^n R(\hat{x}_n) - \textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right]\right)^2\right] \\
&= \beta^{2n}\mathbb{E}_x[R^\pi(\hat{x}_0) - R(\hat{x}_n)]^2 \\
&\quad + \mathbb{E}[(\beta^n R(\hat{x}_n) - \mathbb{E}[\beta^n R(\hat{x}_n)])^2] \\
&\quad + \mathbb{E}[\left(\textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t) - \mathbb{E}\left[\textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right]\right)^2] \\
&= \beta^{2n}\mathbb{E}_x[R^\pi(\hat{x}_n) - R(\hat{x}_n)]^2 + \beta^{2n}\mathbb{V}(R(\hat{x}_n)) + \mathbb{V}\left(\textstyle\sum_{t=0}^{n-1} \beta^t r(\hat{x}_t)\right).
\end{aligned}$$

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. □

Remark 107. *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 1error 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 108. *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 preform 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

$$R(x) \overset{\alpha}{\leftarrow} \sum_{n=0}^{\infty}(1-\lambda)\lambda^n \sum_{k=0}^{n}\beta^k d(\hat{x}_k, \hat{x}_{k+1}) = \sum_{k=0}^{\infty}\lambda^k\beta^k d(\hat{x}_k\hat{x}_{k+1})$$

for each x. Here we let $\hat{x}_0$ be the first visit to x and we let $\hat{x}_1, \hat{x}_2, \ldots$ be the subsequent states.

Def 109 (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

$$R(x) \overset{\alpha}{\leftarrow} \sum_{v=1}^{\infty}\sum_{k=0}^{\infty}\lambda^k\beta^k d(\hat{x}_k^{(v)}\hat{x}_{k+1}^{(v)})$$

where $\hat{x}_0^{(v)}$ be the vth visit to x and we let $\hat{x}_1^{(v)}, \hat{x}_2^{(v)}, \ldots$ 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:

$$E(x) \leftarrow (\lambda\beta)E(x) + \alpha \mathbb{I}[x_t = x] \tag{4.3}$$

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 110 (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 111 (TD(λ) – Online)**.** *At each time step with current state x and next state $\hat{x}$, perform the following update to every state x'*

$$E(x') \leftarrow (\lambda\beta)E(x') + \alpha\mathbb{I}[x' = x]$$
$$R(x') \leftarrow R(x') + E(x')d(x, \hat{x}).$$

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 $R(x) = \mathbb{E}_x^P\Big[\sum_t \beta^t r(\hat{x}_t)\Big]$ 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 $f : \mathcal{X}^2 \to \mathbb{R}$. For instance, TD(0) is searching for the fixed point

$$0 = \mathbb{E}_x^P\left[r(x) + \beta R(\hat{x}) - R(x)\right] = \mathbb{E}_x^Q\left[r(x) + \beta R(\hat{x})\frac{P_{x\hat{x}}}{Q_{x\hat{x}}} - R(x)\right].$$

Thus, given the simulator generates transitions under Q, an importance sampled TD(0) w.r.t. P would be

$$R(x) \overset{\alpha}{\leftarrow} r(x) + \beta R(\hat{x})\frac{P_{x\hat{x}}}{Q_{x\hat{x}}} - R(x).$$

Notice importance sampling for more general functions would be,

$$\mathbb{E}_x^P[f(\hat{x}_0, ..., \hat{x}_t)] = \mathbb{E}_x^Q\left[f(\hat{x}_0, ..., \hat{x}_t)\prod_{s=0}^{t-1}\frac{P_{\hat{x}_s,\hat{x}_{s+1}}}{Q_{\hat{x}_s,\hat{x}_{s+1}}}\right].$$

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

$$R(\hat{x}_0) \overset{\alpha}{\leftarrow} r(\hat{x}_0) + ... + \beta^{\sigma-1} r(\hat{x}_{\sigma-1}) + \beta^{\sigma} R(\hat{x}_\sigma) - R(\hat{x}_0)$$

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 $\hat{x}^{(1)}_{\tau+1}, \hat{x}^{(1)}_{\tau+2}, ..., \hat{x}^{(1)}_{\sigma^{(1)}}$ and $\hat{x}^{(2)}_{\tau+1}, \hat{x}^{(2)}_{\tau+2}, ..., \hat{x}^{(2)}_{\sigma^{(2)}}$ are two trajectories for stopping times $\sigma^{(1)}$ and $\sigma^{(2)}$ after time τ. We can then perform two TD updates

$$R(\hat{x}_0) \overset{\alpha}{\leftarrow} \sum_{t=0}^{\tau} \beta^t r(\hat{x}_t) + \sum_{t=\tau+1}^{\sigma^{(1)}-1} \beta^{\tau+1} r(\hat{x}^{(1)}_{\tau+1}) + \beta^{\sigma^{(1)}} R(\hat{x}^{(1)}_{\sigma^{(1)}}) - R(\hat{x}_0)$$

$$R(\hat{x}_0) \overset{\alpha}{\leftarrow} \sum_{t=0}^{\tau} \beta^t r(\hat{x}_t) + \sum_{t=\tau+1}^{\sigma^{(2)}-1} \beta^{\tau+1} r(\hat{x}^{(2)}_{\tau+1}) + \beta^{\sigma^{(2)}} R(\hat{x}^{(2)}_{\sigma^{(2)}}) - R(\hat{x}_0)\,.$$

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 [38]. So the text of Sutton and Barto is the best place to go to to read more

on this [39].

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 3.1 when proving convergence.

A key ingredient is the notion of a Q-factor as described in Section 1.4. 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 30 we showed that this solved the recursion:

$$Q(x,a) = \mathbb{E}_{x,a}[r(x,a) + \beta \max_{\hat{a}} Q(\hat{X},\hat{a}))]\,. \tag{4.4}$$

Def 112 (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*

$$Q(x,a) \overset{\alpha}{\leftarrow} r(x,a) + \beta \max_{a' \in \mathcal{A}} Q(\hat{x},a') - Q(x,a)$$

where α positive (learning rate) parameter. Recall $x \overset{\alpha}{\leftarrow} dx$ 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.

Thrm 113. *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

$$\sum_{t=1}^{\infty} \alpha_t(x,a) = \infty, \qquad \sum_{t=1}^{\infty} \alpha_t^2(x,a) < \infty$$

then, with probability 1,

$$Q_t(x,a) \to Q^*(x,a)$$

where $Q^*(x,a)$ *is the optimal value function.*

Proof. We essentially show that the result is a consequence of Theorem 7 in Section 3.1. We note that the optimal Q-function, $\boldsymbol{Q} = (Q(x,a) : x \in \mathcal{X}, a \in \mathcal{A})$ satisfies a fixed point equation

$$\boldsymbol{Q} = \boldsymbol{F}(\boldsymbol{Q}),$$

with

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

for each $x \in \mathcal{X}$ and $a \in \mathcal{A}$. We know from Prop 30 that for discounted programming $\boldsymbol{F}(\cdot)$ is a contraction. I.e.

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

Now notice that the Q-learning algorithm performs the update

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha_t(x,a)(F(\boldsymbol{Q})(x,a) - Q_t(x,a) + \epsilon(x,a)),$$

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 7 and that the condtions of that theorem are satisfied. Thus it must be that

$$Q_t(x,a) \xrightarrow[t\to\infty]{} Q^*(x,a)$$

where $Q^*(x,a)$ satisfies $\boldsymbol{Q}^* = \boldsymbol{F}(\boldsymbol{Q}^*)$. In otherwords, as required, it satisfies the Bellman equation for the optimal Q-function and thus is optimal. □

[2]Note $(x+y)^2 \leq 2x^2 + 2y^2$

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:

$$Q^{(A)}(x,a) \overset{\alpha}{\leftarrow} r(x,a) + \beta Q^{(A)}(\hat{x},\hat{b}) - Q^{(A)}(x,a)$$
$$Q^{(B)}(x,a) \overset{\alpha}{\leftarrow} r(x,a) + \beta Q^{(B)}(\hat{x},\hat{a}) - Q^{(B)}(x,a)$$

[3]The 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 $\mathbb{E}[A_{\hat{b}} t] = \bar{A}_{\hat{b}}$ (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 114 (Advantage)**.** *For a value function $V(x)$ and Q-factors $Q(x,a)$, the advantage function is defined by*

$$A(x,a) := Q(x,a) - V(x), \qquad x \in \mathcal{X}, a \in \mathcal{A}.$$

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:

$$A(x,a) \leftarrow A(x,a) - A_{\max}(x) \tag{4.5a}$$

$$A(x,a) \overset{\alpha}{\leftarrow} r + \beta V(\hat{x}) - V(x) + A_{\max}(x) - A(x,a) \tag{4.5b}$$

$$V(x) \overset{\gamma}{\leftarrow} \Delta A_{\max}(x)/\alpha \tag{4.5c}$$

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 3. *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:

$$\begin{aligned} Q(x) &= V(x) + A(x,a) - A_{\max}(x) \\ &\leftarrow V(x) + A(x,a) - A_{\max}(x) \\ &\quad + \alpha\left(r + \beta V(x) - V(x) - A(x,a) + A_{\max}(x)\right) \\ &= Q(x,a) + +\alpha\left(r + \beta \max_a Q(x,a) - Q(x,a)\right). \end{aligned}$$

Q-learning converges under the appropriate step size choice given in Theorem 113. So since $Q(x,a)$ convergence and $V(x) = \max_a Q(x,a)$, $V(x)$ convergences. 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. □

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 [40]. An alternative proof is given by Jaakkola et al.[22] which applies a slightly esoteric fixed point method of Dvoretzky.

Double Q-learning is first proposed by Hasselt [20]. The Advantage updating algorithm [with some minor modification to (4.5a)] is first given by Baird [2]. Both ideas have been successfully applied with neural network function approximation [43, 44].

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 115 (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*

$$Q(x,a) \overset{\alpha}{\leftarrow} r(x,\hat{x}) + \beta Q(\hat{x},\hat{a}) - Q(x,a)$$

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 116 (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 9. *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

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha_t(x,a)\left\{r(x_t,a_t) + \beta Q_t(x_{t+1},a_{t+1}) - Q_t(x_t,a_t)\right\}$$

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

$$\begin{aligned}
&Q_{t+1}(x_t, a_t) \\
&= Q_t(x_t, a_t) + \alpha_t(x_t, a_t)\left\{r(x_t, a_t) - \beta \max_{\hat{a}} Q_t(x_{t+1}, \hat{x}) - Q_t(x_t, a_t)\right\} \\
&\quad + \alpha_t(x_t, a_t)\beta\left[\max_{\hat{a}} Q_t(x_{t+1}, \hat{x}_- Q_t(x_{t+1}, a_{t+1})\right] \\
&= Q_t(x_t, a_t) + \alpha_t(x_t, a_t)\left[F_{x_t,a_t}(\boldsymbol{Q}_t) - Q_t(x_t, a_t)\right] + \alpha_t(x_t, a_t)\delta_t + \alpha_t(x_t, a_t)\beta e_t
\end{aligned}$$

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

$$\begin{aligned}
\delta_t :=& r(x_t, a_t) + \beta \max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q(x,a) - \mathbb{E}[r(x_t, a_t) + \beta \max_{\hat{a}} Q(x_t, \hat{a}) - Q(x_t, a_t)|\boldsymbol{F}_t] \\
&+ \max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1}) - \mathbb{E}[\max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1})|\boldsymbol{F}_t]
\end{aligned}$$

where here $\boldsymbol{F}_t = (x_s, a_s : s \leq t)$ and we note that δ_t is a bounded Martingale difference sequence; and

$$e_{t+1} = \mathbb{E}\left[\max_{\hat{a}} Q_t(x_{t+1}, \hat{a}) - Q_t(x_{t+1}, a_{t+1})\middle|\boldsymbol{F}_t\right]$$

which satisfies

$$e_t \to 0$$

as $t \to \infty$, since $Q_t(x,a)$ is bounded (note each update is bounded for a discounted progrma) and our policy is GLIE (so the probably of choosing the maximizing action goes to one).

Thus if we define $\epsilon_t = \delta_t + \beta e_t$, then we safisfy exactly the conditions of the Asynchronous Robbins-Munro update 3.5 in Theorem 7, that is

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha_t(x,a)\left[F_{x,a}(\boldsymbol{Q}_t) - Q_t(x,a) + \epsilon_t\right]$$

and by that result, for all x, a,

$$Q_t(x,a) \xrightarrow[t\to\infty]{} Q^*(x,a)$$

where $\boldsymbol{Q}^\star = F(\boldsymbol{Q}^\star)$ or, in other words,

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

So the Bellman equaiton is satisfied and so the limit $Q^\star(x,a)$ is the optimal value function. □

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 117. *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

Function Approximation

5.1 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 $f : \mathcal{X} \to \mathcal{Y}$ 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 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)}) : n = 1, ..., N\}$ and a set of functions $\mathcal{F}$ our goal is to selcted 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

$$\min_{g \in \mathcal{F}} \mathbb{E}\left[L(\hat{y}, g(\hat{x}))\right]$$

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.

5.2 Linear Regression

Using data $(\boldsymbol{x}^{(n)}, y^{(n)})$, $n = 1, ..., N$, you want to approximate a real-valued output variable y from a p-dimensional input vector $\boldsymbol{x}$. We

approximate y by a linear function of $\boldsymbol{x}$, namely,

$$\boldsymbol{\theta}^\top \boldsymbol{x} = \theta_0 + \theta_1 x_1 + ... + \theta_K x_K \ .$$

Here $\boldsymbol{x} = (1, x_1, ..., x_p)$. We choose the $\boldsymbol{\theta}$ that gives the least square distance between y and $\boldsymbol{\theta}^\top \boldsymbol{x}$ for each data point:

$$\text{minimize} \quad \sum_{n=1}^{N} \left(y^{(n)} - \boldsymbol{\theta}^\top \boldsymbol{x}^{(n)} \right)^2 \quad \text{over} \quad \boldsymbol{\theta} \in \mathbb{R}^{p+1} \, .$$

Optimal Weights. A short calculation gives that this minimization is solved by

$$\hat{\boldsymbol{\theta}} = (X^\top X)^{-1} X^\top \boldsymbol{y}$$

where we define the $p \times p$ matrix X, by $X_{ij} = \sum_n x_i^{(n)} x_j^{(n)}$ and we define $\boldsymbol{y} = (y^{(1)}, ..., y^{(N)})^\top$. 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:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \left(y^{(n)} - \boldsymbol{\theta}^\top \boldsymbol{x}^{(n)} \right) \boldsymbol{x}^{(n)} \, .$$

So we move $\boldsymbol{\theta}$ in the direction of $\boldsymbol{x}^{(n)}$, 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 $\boldsymbol{x}$, nor does $\boldsymbol{x}$ need to be finite dimensional, but we do need a linear relationship with respect to $\boldsymbol{\theta}$.

Suppose for each $\boldsymbol{x}$ we take p different *features* using the *basis functions* $\phi_j(\boldsymbol{x})$, $j = 1, ..., p$. We can then perform linear regression on

$$\boldsymbol{\theta}^\top \boldsymbol{\phi} = \theta_0 + \theta_1 \phi_1(\boldsymbol{x}) + ... + \theta_p \phi_p(\boldsymbol{x}) \, .$$

The least squares distance is given by

$$\hat{\boldsymbol{\theta}} = (\Phi^\top \Phi)^{-1} \Phi^\top \boldsymbol{y}$$

where we define the $p \times p$ matrix Φ, by $\Phi_{ij} = \sum_n \phi_i(\boldsymbol{x}^{(n)}) \phi_j(\boldsymbol{x}^{(n)})$.

Polynomial Regression. A good example is polynomial regression. Here we can model y as a polynomial function of x, with the basis functions $\phi_0(x) = 1, \phi_1(x) = x, \phi_2(x) = x^2, ..., \phi_p(x) = x^p$.

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:

$$\text{minimize} \quad \sum_{n=1}^{N} \left(y^{(n)} - \boldsymbol{\theta}^\top \boldsymbol{x}^{(n)}\right)^2 + \underbrace{\lambda \|\boldsymbol{\theta}\|_2^2}_{\text{penalty term}} \qquad \text{over} \quad \boldsymbol{\theta} \in \mathbb{R}^{p+1}\,.$$

where $\|\boldsymbol{\theta}\|_2^2 = \sum_j \theta_j^2$ is the L_2 norm. This is know as *ridge regression* or L_2-regularization.[1] We could use the L_1 norm

$$\|\boldsymbol{\theta}\|_1 = \sum_j |\theta_j|.$$

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

$$\hat{\boldsymbol{\theta}} = (X^\top X + \lambda I)^{-1} X^\top \boldsymbol{y}$$

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

$$\boldsymbol{\theta} \leftarrow (1 - \alpha\lambda)\boldsymbol{\theta} - \alpha\left(y^{(n)} - \boldsymbol{\theta}^\top \boldsymbol{x}^{(n)}\right)\boldsymbol{x}^{(n)}\,.$$

Essentially the $(1 - \alpha\lambda)$ term means we apply a dampening effect on the stochastic gradient descent update.

[1] We 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 $\boldsymbol{\theta}$ 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

$$\begin{aligned} \text{minimize} \quad & \sum_{n=1}^{N} \left(y^{(n)} - \boldsymbol{\theta}^\top \boldsymbol{x}^{(n)} \right)^2 \\ \text{subject to} \quad & \|\boldsymbol{\theta}\|_2^2 \leq \kappa \\ \text{over} \quad & \boldsymbol{\theta} \in \mathbb{R}^{p+1} . \end{aligned}$$

In this way we can think of regularization as reducing the complexity of our model. As the number of models that we are considering if reduced.

5.3 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

$$\hat{\mathbb{E}}_{\mathcal{D}}[L(\hat{y}, f(\hat{\boldsymbol{x}}))] := \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} L(y, x) .$$

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 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)}) : n = 1, ...N\}$ into a *training set*, $\mathcal{D}_{\text{train}}$, a *development (dev) set* $\mathcal{D}_{\text{dev}}$ and a *test set*, $\mathcal{D}_{\text{train}}$. 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 alway simulate new data for training, developing, testing.

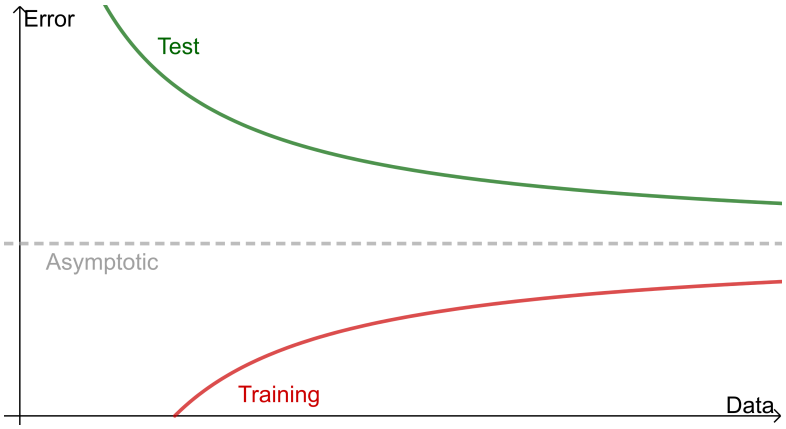


Figure 5.1: Traing Error (red) and Test Error (green) approach the Asymptotic Error as Data increases.

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:

$$\begin{aligned} \text{Training Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{train}}}[L(\hat{y}, f(\hat{x}))] \\ \text{Dev Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{dev}}}[L(\hat{y}, f(\hat{x}))] \\ \text{Test Error} &= \hat{\mathbb{E}}_{\mathcal{D}_{\text{train}}}[L(\hat{y}, f(\hat{x}))] \\ \text{Asymptotic Error} &= \min_{\theta} \mathbb{E}\left[L(\hat{y}, f(\hat{x}))\right] \end{aligned}$$

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 is 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 5.1. 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.

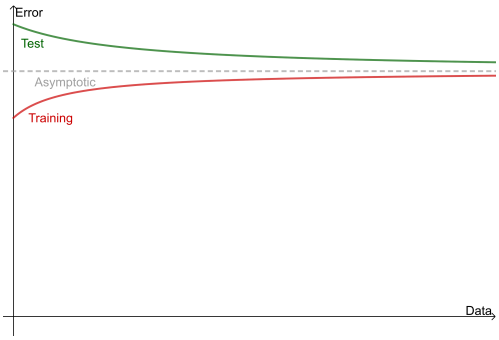


Figure 5.2: A Simple Model

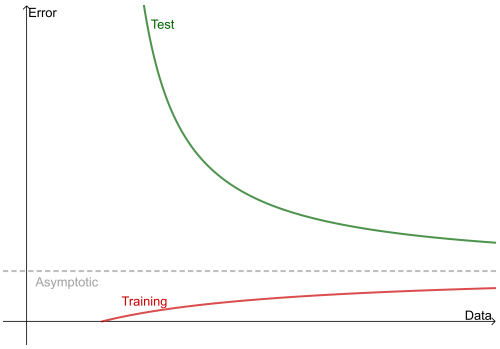


Figure 5.3: A Complex Model

5.4 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

$$\text{bias}_x(\hat{f}) = \left(\bar{f}(x) - f(x)\right)^2 \qquad \text{var}_x(\hat{f}) = \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}(x) - \bar{f}(x)\right)^2\right].$$

We can also take an expectation over the distribution of $\hat{x}$ to get the bias and variance of $\hat{f}$. I.e.

$$\text{bias}(\hat{f}) = \mathbb{E}_{\hat{x}}\left[\left(\bar{f}(\hat{x}) - f(\hat{x})\right)^2\right], \qquad \text{var}(\hat{f}) = \mathbb{E}_{\mathcal{D},\hat{x}}\left[\left(\hat{f}(\hat{x}) - \bar{f}(\hat{x})\right)^2\right].$$

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 12 (Bias-Variance Decomposition)**.**

$$\mathbb{E}_{(\hat{x},\hat{y}),\mathcal{D}}\left[\left(\hat{y} - \hat{f}(\hat{x})\right)^2\right] = \textit{bias}(\hat{f}) + \textit{var}(\hat{f}) + \textit{var}(\epsilon).$$

Proof. Since $\hat{y} = f(\hat{x}) + \epsilon$, we can expand as follows

$$\begin{aligned}
\mathbb{E}\left[\left(\hat{y} - \hat{f}\right)^2\right] =& \mathbb{E}\left[\left(\epsilon + f - \bar{f} + \bar{f} - \hat{f}\right)^2\right] \\
=& \underbrace{\mathbb{E}[\epsilon^2]}_{var(\epsilon)} + \underbrace{2\mathbb{E}[\epsilon]\mathbb{E}[(f - \bar{f} + \bar{f} - \hat{f})]}_{=0,\text{ independence and } E[\epsilon]=0} \\
&+ \underbrace{\left(f - \bar{f}\right)^2}_{\text{bias}_x(\hat{f})} + 2\left(f - \bar{f}\right)\underbrace{\mathbb{E}\left[\left(\bar{f} - \hat{f}\right)\right]}_{=0} + \underbrace{\mathbb{E}\left[\left(\bar{f} - \hat{f}\right)^2\right]}_{\text{var}_x(\hat{f})}. \\
=& \text{bias}(\hat{f}) + \text{var}(\hat{f}) + \text{var}(\epsilon).
\end{aligned}$$

□

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 5.2 and 5.3 and take a cross section where the amount of data is fixed. We get a picture like Figure 5.4.

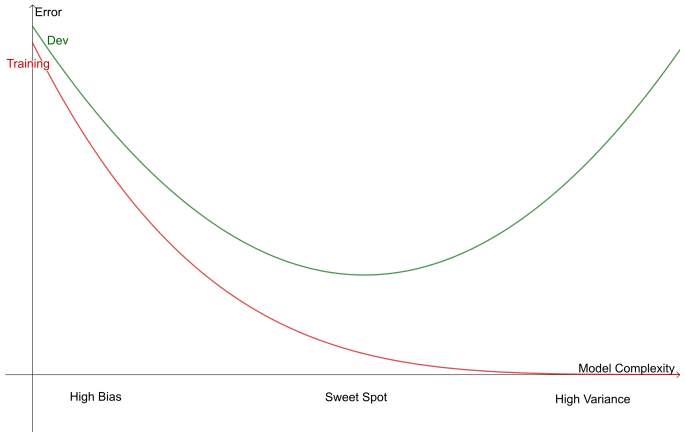


Figure 5.4: Training and test error for fixed amount of data.

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 5.1 and Figure 5.4 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])

[2]These 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 5.1: Table of error symptoms, diagnosis and remdey for learning problems.

Diagnostics. We with 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 potential up the bias). If there is no more data, increasing regularization parameter is a good alternative.

conceptually easy Bias-Variance calculation into the 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 regularlize 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.

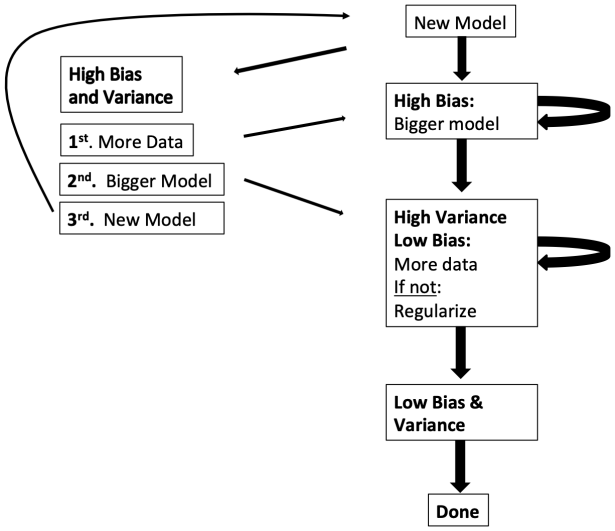


Figure 5.5: Flow chart for getting a working machine learning model.

5.5 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 $\boldsymbol{x} = (1, x_1, ..., x_p)$ a weight $\boldsymbol{w}$ is applied to each of these and then this is applied to a function to give our prediction:

$$\hat{y} = f(\boldsymbol{w}^\top \boldsymbol{x})$$

The function $f(z)$ is often called an activation function. This is represented in Figure 5.6

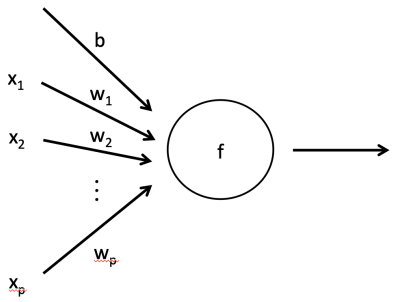


Figure 5.6: A Neuron in a Neural network.

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:

$$\frac{\partial \bar{y}}{\partial w_j} = x_j f'(\boldsymbol{w}^\top \boldsymbol{x}) \,.$$

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 5.2

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 5.2: 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 5.7 for an instance of a two-layer neural network.

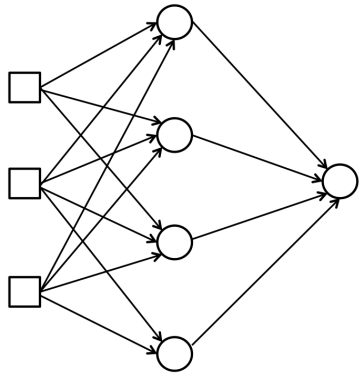


Figure 5.7: A two-layer neural network. Here squares give inputs and circles are neurons.

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 $\boldsymbol{x}$ and weights $\boldsymbol{w}$ 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 $\boldsymbol{w}^{(k)}$ 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, $l = 1, ..., l'$. (Meaning, roughly, between 3 and 300 layers). Here a layer, l, is a set of activation functions $f_i^{(l)}$, $i = 1, ..., n_l$, and each activation receives, as input, the output from the previous layer. These are called Deep Neural Networks. See Figure 5.8.

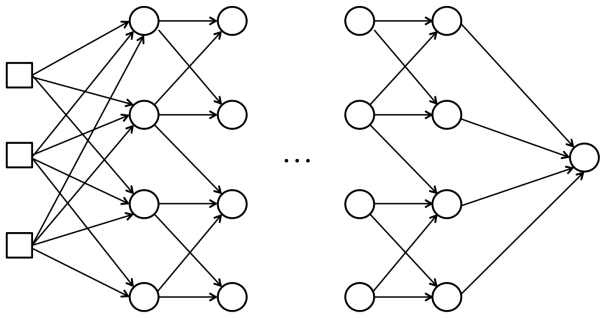


Figure 5.8: A deep neural network.

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 $(\boldsymbol{x}, y)$ i.e. an input vector $\boldsymbol{x}$ and an output number y. We can get a prediction $\bar{y}$ from $\boldsymbol{x}$ by recursively applying our above formula:

$$a_j^{(l)} = f_j^{(l)}\Big(\sum_i a_i^{(l-1)} w_{ij}^{(l)}\Big). \tag{5.1}$$

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 (5.1), we must apply backward a related formula. Hence this takes its name as "backpropagation". 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 $z_j^{(l)}$:

$$\begin{aligned}\delta_j^{(l)} &= \sum_k \frac{\partial z_k^{(l+1)}}{\partial z_j^{(l)}} \frac{\partial L}{\partial z_k^{(l+1)}} \\ &= \sum_k \dot{a}_j^{(l)} w_{jk}^{(l+1)} \delta_k^{(l+1)}\end{aligned} \tag{5.2}$$

Notice once we have obtained the values of each $z_j^{(l)}$ in our forward pass using (5.1) , we can the work backwards using (5.2) 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 5.9. The key terms and formulas in backpropagation are summarized in Table 5.3, below.

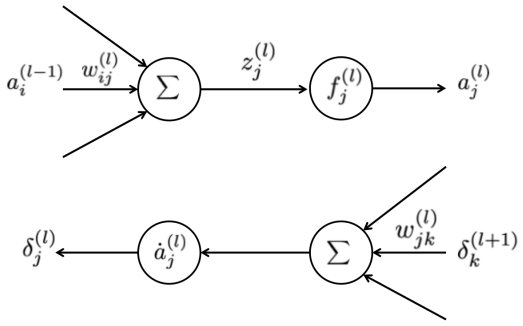


Figure 5.9: Forward and backward pass of Backpropagation.

References

Everything on statistical learning, linear regression, bias-variance and neural networks is by now *very* standard machine learning. See Friedman et al. [15], Murphy [30] or Goodfellow [17].

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 5.3: Backpropagation Definitions.

Chapter 6

Reinforcement Learning with Function Approximation

6.1 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 $\hat{\boldsymbol{x}} = (\hat{x}_t : t \in \mathbb{Z}_+)$, consider the reward function

$$R(x) := \mathbb{E}_x\left[\sum_{t=0}^{\infty} r(\hat{x}_t)\right] \tag{6.1}$$

associated with rewards given by $\boldsymbol{r} = (r(x) : x \in \mathcal{X})$. We approximate the reward function $R(x)$ with a linear approximation,

$$R(x; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{\phi}(x) = \sum_{j \in \mathcal{J}} w_j \phi_j(x).$$

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 $\boldsymbol{w} = (w_j : j \in \mathcal{J})$ 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

$$R(x) = \mathbb{E}_x[\underbrace{r(x) + \beta R(\hat{x})}_{=:\text{Target}(x)}], \qquad x \in \mathcal{X}. \tag{6.2}$$

The target, $\text{Target}(x)$, is an estimate of the true value of $R(x; \boldsymbol{w})$ Here the target random variable considered is the TD(0) target. Other targets can be used, e.g. the term in the sum, (6.1), 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

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \mathbb{E}_\mu\left[(\text{Target}(x) - R(x; \boldsymbol{w}))^2\right].$$

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 μ. We can only get samples

and so we can instead apply Robbin's-Munro/Stochastic Gradient Descent update to $\boldsymbol{w}$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\Big(\text{Target}(x) - R(x;\boldsymbol{w})\Big)\nabla_{\boldsymbol{w}} R(x;\boldsymbol{w})\,.$$

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 $r(x)+\beta R(\hat{x};\boldsymbol{w})$, where $\hat{x}$ is the next state after x. Under a linear function approximation this gives an update

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\Big(r(x) + \beta\boldsymbol{w}^\top\boldsymbol{\phi}(\hat{x}) - \boldsymbol{w}^\top\boldsymbol{\phi}(x)\Big)\boldsymbol{\phi}(x)\,.$$

we let

$$\boldsymbol{g}(x;\boldsymbol{w}) := \Big(r(x) + \beta\boldsymbol{w}^\top\boldsymbol{\phi}(\hat{x}) - \boldsymbol{w}^\top\boldsymbol{\phi}(x)\Big)\boldsymbol{\phi}(x)\,.$$

Convergence Result. We argue (informally) that, for this iteration scheme, $w(t) \to w^*$ where the limit function Φw^* is in a factor of the best approximation

$$\|\boldsymbol{R} - \boldsymbol{R}(\boldsymbol{w}^*)\|_\mu \le \frac{1}{1-\beta}\|\boldsymbol{R} - \Pi(\boldsymbol{R})\|_\mu\,.$$

Here we interpret $\boldsymbol{R}$ and $\boldsymbol{R}(\boldsymbol{w}^*)$ as vectors $\boldsymbol{R} = (R(x) : x \in \mathcal{X})$ and $\boldsymbol{R}(\boldsymbol{w}) = (R(x;\boldsymbol{w}) : x \in \mathcal{X})$. Also, we define the norm above by

$$\|\boldsymbol{R}\|_\mu^2 = \sum_{x\in\mathcal{X}} \mu(x)R(x)^2\,.$$

Average Behaviour. Suppose that our Markov chain $\hat{\boldsymbol{x}}$ is stationary with stationary distribution $\mu(x)$. If we look at the expected change in our update term we get

$$\begin{aligned}
&\mathbb{E}_\mu[g(x;\boldsymbol{w})] \\
&= \sum_x \mu(x)r(x)\boldsymbol{\phi}(x) + \sum_x\sum_y \mu(x)(\beta\boldsymbol{\phi}(x)P_{xy}\boldsymbol{\phi}(y)^\top - \boldsymbol{\phi}(x)\boldsymbol{\phi}(x)^\top)\boldsymbol{w} \\
&= \Phi^\top M\boldsymbol{r} - \Phi^\top M[I-\beta P]\Phi\boldsymbol{w}\,. \qquad (6.3)
\end{aligned}$$

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 $\boldsymbol{b}$ and the $\mathcal{J} \times \mathcal{J}$ matrix A, as defined follows:

$$A := \Phi^\top M[I - \beta P]\Phi \qquad \text{and} \qquad b := \Phi^\top M\boldsymbol{r}\,.$$

Differential Equation Analysis. So, roughly, $\boldsymbol{w}$ moves according to the differential equation

$$\frac{d\boldsymbol{w}}{dt} = -A\boldsymbol{w} + \boldsymbol{b}\,.$$

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 $(\Phi\boldsymbol{w})^\top M(I - \beta P)\Phi\boldsymbol{w} < -(1 - \beta)\|\Phi\boldsymbol{w}\|_\mu^2$. This then implies that

$$\boldsymbol{v}^\top A\boldsymbol{v} \geq (1 - \beta)\|\Phi\boldsymbol{v}\|_\mu^2.$$

This is proven in Lemma 4 below.

This is sufficient to give convergence of the above differential equation: take $\boldsymbol{w}^*$ such that $A\boldsymbol{w}^* = \boldsymbol{b}$ and take $L(\boldsymbol{w}) = \frac{1}{2}\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|^2$ then

$$\frac{dL}{dt} = \nabla L(\boldsymbol{w}) \cdot \frac{d\boldsymbol{w}}{dt} = -(\boldsymbol{w} - \boldsymbol{w}^*)^\top A(\boldsymbol{w} - \boldsymbol{w}^*) \leq -(1 - \beta)\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu^2.$$

Thus we see that $\boldsymbol{R}(\boldsymbol{w}(t)) = \Phi\boldsymbol{w}(t) \to \Phi\boldsymbol{w}^* = \boldsymbol{R}(\boldsymbol{w}^*)$, and since we assume ϕ_j are linearly independent $\boldsymbol{w}(t) \to \boldsymbol{w}^*$.

Approximation Error. Convergence is great and everything, but we must verify that the solution obtained, $\boldsymbol{w}^*$, is a "good" solution. First, notice that the reward function $\boldsymbol{R} = (R(x) : x \in \mathcal{X})$ satisfies

$$\boldsymbol{R} = T_0(\boldsymbol{R}), \qquad \text{where} \qquad T_0(\boldsymbol{R}) := \boldsymbol{r} + \beta P\boldsymbol{R}\,.$$

This is just (6.2) with $\boldsymbol{R}$ interpreted as a vector and the expectation as a matrix operation with respect to transition matrix P.

Second, notice the approximation $\boldsymbol{R}(\boldsymbol{w}) = (R(x;\boldsymbol{w}) : x \in \mathcal{X})$ that is closest to the rewards $\boldsymbol{R}$, is given by a projection, specifically

$$\Pi(\boldsymbol{R}) := \Phi(\Phi^\top M\Phi)^{-1}\Phi^\top M\boldsymbol{R} = \operatorname*{argmin}_{\boldsymbol{R}(\boldsymbol{w})} \|\boldsymbol{R} - \boldsymbol{R}(\boldsymbol{w})\|_\mu^2\,.$$

Third, we see the equations satisfied by $\boldsymbol{w}^*$ can be expressed in a form somewhat similar to above expression $\boldsymbol{R} = T_0(\boldsymbol{R})$. Specifically, we rearrange the expression $A\boldsymbol{w}^* = b$

$$\begin{aligned} A\boldsymbol{w}^* = \boldsymbol{b} &\iff \Phi^\top M[I - \beta P]\Phi\boldsymbol{w}^* = \Phi^\top M\boldsymbol{r} \\ &\iff \Phi^\top M\Phi\boldsymbol{w}^* = \Phi^\top M\boldsymbol{r} + \Phi^\top M\beta P\Phi\boldsymbol{w}^* \\ &\iff \underbrace{\Phi\boldsymbol{w}^*}_{\boldsymbol{R}(\boldsymbol{w}^*)} = \underbrace{\Phi[\Phi^\top M\Phi]^{-1}\Phi^\top M}_{\Pi}\underbrace{(\boldsymbol{r} + \beta P\Phi\boldsymbol{w}^*)}_{T_0(\boldsymbol{R}^*(\boldsymbol{w}))} . \end{aligned}$$

So, while $\boldsymbol{R}$ satisfies $\boldsymbol{R} = T_0(\boldsymbol{R})$, we see that $\boldsymbol{R}(\boldsymbol{w}^*)$ satisfies

$$\boldsymbol{R}(\boldsymbol{w}^*) = \Pi(T_0(\boldsymbol{R}(\boldsymbol{w}^*)) .$$

We can use these identities satisfied by $\boldsymbol{R}$ and $\boldsymbol{R}(\boldsymbol{w}^*)$ to show that approximation is comparable to the best approximation of $\boldsymbol{R}$. Since T_0 is a β-contraction and that projections always move distances closer (both properties are relatively easy to verify, see Lemma 4):

$$\begin{aligned} \|\boldsymbol{R} - \boldsymbol{R}(\boldsymbol{w}^*)\|_\mu &\le \|\boldsymbol{R} - \Pi(\boldsymbol{R})\|_\mu + \|\Pi(T_0(\boldsymbol{R})) - \Pi(T_0(\boldsymbol{R}(\boldsymbol{w}^*))\|_\mu \\ &\le \|\boldsymbol{R} - \Pi(\boldsymbol{R})\|_\mu + \beta\|\boldsymbol{R} - \boldsymbol{R}(\boldsymbol{w}^*)\|_\mu . \end{aligned}$$

So

$$\|\boldsymbol{R} - \boldsymbol{R}(\boldsymbol{w}^*)\|_\mu \le \frac{1}{1-\beta}\|\boldsymbol{R} - \Pi(\boldsymbol{R})\|_\mu .$$

Some Formal Analysis. Here are a few formal results that we mention in the discussion above.

Proposition 4. *We define* $\|R\|_\mu = \sum_x \mu(x)R(x)^2$ *and* $\|P\|_\mu = \sup_f \|Pf\|_\mu/\|f\|_\mu$
a) $\|P\|_\mu \le 1$.
b) The $TD(0)$ *map is a contraction that is if for function* $R : \mathcal{X} \to \mathcal{X}$ *we let,* $T\boldsymbol{R}(x) = r(x) + \beta P\boldsymbol{R}(x)$ *then*

$$\|T\boldsymbol{R}_1 - T\boldsymbol{R}_2\|_\mu \leqslant \beta\,\|\boldsymbol{R}_1 - \boldsymbol{R}_2\|_\mu$$

and

$$\|\Pi T\boldsymbol{R}_1 - \Pi T\boldsymbol{R}_2\|_\mu \leqslant \beta\,\|\boldsymbol{R}_1 - \boldsymbol{R}_2\|_\mu$$

where Π *is a projection in* $\|\cdot\|_\mu$.
c)

$$(w - w^*)^\top\,\mathbb{E}_\mu[g(x;w)] \le -(1-\beta)\,\|\Phi w - \Phi w^*\|_\mu^2 .$$

Proof. a) Using Jensen's Inequality below,

$$\begin{aligned}\|Pf\|_\mu^2 &= \sum_x \mu(x)\Big(\sum_y P_{xy} f(y)\Big)^2 \leq \sum_x \mu(x) \sum_y P_{xy} f(y)^2 \\ &= \sum_y f(y)^2 \underbrace{\sum_x \mu(x) P_{xy}}_{=\mu(y)} = \sum_y \mu(y) f(y)^2 = \|f\|_\mu^2 .\end{aligned}$$

In the curly brace, we are using that μ is a stationary distribution.
b)

$$\|T\boldsymbol{R}_1 - T\boldsymbol{R}_2\|_\mu = \beta \|P(\boldsymbol{R}_1 - \boldsymbol{R}_2)\|_\mu \leqslant \beta \|P\|_\mu \|\boldsymbol{R}_1 - \boldsymbol{R}_2\|_\mu$$

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 (6.3), we have that

$$\begin{aligned}\mathbb{E}_\mu[\boldsymbol{g}(x;\boldsymbol{w})] &= \Phi^T M[r + \beta P - I]\Phi\boldsymbol{w} \\ &= \Phi^\top M\left[T(\Phi\boldsymbol{w}) - \Phi\boldsymbol{w}\right] \\ &= \Phi^\top M[(I - \Pi) + \Pi]\left[T(\Phi\boldsymbol{w}) - \Phi\boldsymbol{w}\right] \\ &= \Phi^\top M\left[\Pi T(\Phi\boldsymbol{w}) - \Phi\boldsymbol{w}\right] .\end{aligned}$$

In the third, equality we use that $\Phi^\top(I - \Pi) = 0$ 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

$$\begin{aligned}&(\boldsymbol{w} - \boldsymbol{w}^*)^\top \mathbb{E}_\mu[g(x;\boldsymbol{w})] \\ &= (\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*)^\top M\left[\Pi T(\Phi\boldsymbol{w}) - \Phi\boldsymbol{w}\right] \\ &= (\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*)^\top M\left[\Pi T(\Phi\boldsymbol{w}) - \Pi T(\Phi\boldsymbol{w}^*) + \Phi\boldsymbol{w}^* - \Phi\boldsymbol{w}\right] \\ &= -\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu^2 + (\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*)^\top M\left[\Pi T(\Phi\boldsymbol{w}) - \Pi T(\Phi\boldsymbol{w}^*)\right] \\ &\leq -\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu^2 + \|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu \|\Pi T(\Phi\boldsymbol{w}) - \Pi T(\Phi\boldsymbol{w}^*)\|_\mu^2 \\ &\leq -\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu^2 + \|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu \beta \|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu \\ &= -(1 - \beta)\|\Phi\boldsymbol{w} - \Phi\boldsymbol{w}^*\|_\mu^2\end{aligned}$$

In the second equality above, we use that $\boldsymbol{w}^*$ satisfied $\Pi T(\Phi\boldsymbol{w}^*) = \Phi\boldsymbol{w}^*$. The first inequality is the Cauchy-Schwartz inequality. The second inequality, applies the contraction property from part b). □

References

The section is based on reading Tsitsiklis and Van Roy [41], but also see Bertsekas and Tsitsiklis [6].

6.2 Linear Approximation and Stopping

We consider an optimal stopping problem as introduced in Section 1.6, and we apply a linear function approximation scheme like in Section 6.1.

It is not straight-forward to prove that the results of Section 6.1 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 done 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 $\hat{\boldsymbol{x}} = (\hat{x}_t : t \in \mathbb{Z}_+)$ 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

$$V(x) = \max_{\tau} \mathbb{E}_x \left[\sum_{t=0}^{\tau-1} \beta^t r(\hat{x}_t) + \beta^\tau \bar{r}(\hat{x}_\tau) \right] \tag{6.4}$$

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 $\boldsymbol{R} = (R(x) : x \in \mathcal{X})$, we let

$$T\boldsymbol{R}(x) = \max \{\bar{r}(x), r(x) + \beta E_x[R(\hat{x})]\} .$$

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 Q-function vector $\boldsymbol{Q}' = (Q'(x) : x \in \mathcal{X})$ by

$$S\boldsymbol{Q}'(x) = r(x) + \beta \mathbb{E}_x \left[\max \{ \bar{r}(\hat{x}), Q'(\hat{x}) \} \right] .$$

The value function V in (6.4) 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 1.6 and Theorem 31 in Section 1.4. 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 $\hat{\boldsymbol{x}} = (\hat{x}_t : t \in \mathbb{Z}_+)$ 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,

$$Q(x; \boldsymbol{w}) = [\Phi \boldsymbol{w}]_x = \boldsymbol{w}^\top \boldsymbol{\phi}(x) = \sum_{j \in \mathcal{J}} w_j \phi_j(x).$$

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 $\boldsymbol{w} = (w_j : j \in \mathcal{J})$ to each of these features. Our job is to find weights so that $Q(x; \boldsymbol{w})$ gives a good approximation to $Q(x)$. We can intepret the Q-function and approximation as vectors with components in $\mathcal{X}$ given by $\boldsymbol{Q} = (Q(x) : x \in \mathcal{X})$ and $\boldsymbol{Q}_{\boldsymbol{w}} = \Phi \boldsymbol{w}$ where Φ is a matrix whose x-th column is given by $\boldsymbol{\phi}(x) = (\phi_j(x) : j \in \mathcal{J})$. We we can use this to define a stopping policy given by

$$\tau(\boldsymbol{w}) = \min \{ t : \bar{r}(x) \geq Q(x; \boldsymbol{w}^*) \} .$$

Since $\tau(\boldsymbol{w})$ does not necessarily induce the same Q-function as $Q(x; \boldsymbol{w}^*)$. We will later consider the operation

$$S_{\boldsymbol{w}} Q(x) := r(x) + \beta \mathbb{E}_x [r(\hat{x}) \mathbb{I}[r(x) \geq \Phi \boldsymbol{w}(x)] + Q(x) \mathbb{I}[r(x) < \Phi \boldsymbol{w}(x)]]$$

Think of $S_{\boldsymbol{w}}$ being the value of following stopping rule $\tau(\boldsymbol{w})$ on the next step and there afterward following Q. If is immediate that

$$S_{\boldsymbol{w}} \Phi \boldsymbol{w} = S \Phi \boldsymbol{w}$$

As before it is useful to consider the projection onto the space spanned by the basis functions Φ, namely,

$$\Pi(\boldsymbol{Q}) := \Phi(\Phi^\top M\Phi)^{-1}\Phi^\top M\boldsymbol{Q} = \operatorname*{argmin}_{\boldsymbol{Q}_w} \|\boldsymbol{Q} - \boldsymbol{Q}_w\|_\mu^2 \tag{6.5}$$

where as before $\|\boldsymbol{Q}\|_\mu^2 = \sum_{x\in\mathcal{X}} \mu(x)Q(x)^2$ 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

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t\boldsymbol{\phi}(x_t)\Big(r(x_t) + \beta\max\{Q(x_{t+1};\boldsymbol{w}_t), \bar{r}(x_{t+1})\} - Q(x_t;\boldsymbol{w}_t)\Big) \tag{6.6}$$

here α_t is a step size parameter. Notice this is similar the TD(0) update given in Section 6.1.

If we let

$$\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{\phi}(x_t)\left(r(x_t) + \beta\max\{Q(x_{t+1};\boldsymbol{w}_t), \bar{r}(x_{t+1})\} - Q(x_t;\boldsymbol{w}_t)\right)$$

Then the update (6.6) is simply, $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t\boldsymbol{g}(\boldsymbol{w}_t)$.

Average Behavior. If we look at the expected change in $\boldsymbol{g}(\boldsymbol{w})$ given $x_t = x$, we get that

$$\begin{aligned}\mathbb{E}_x[\boldsymbol{g}(\boldsymbol{w})] &= \boldsymbol{\phi}(x)\left(r(x) + \beta\sum_y P_{xy}\max\left\{\boldsymbol{\phi}(y)^\top\boldsymbol{w}, \bar{r}(y)\right\} - \boldsymbol{\phi}(x)^\top\boldsymbol{w}\right)\\ &= \boldsymbol{\phi}(x)\left(SQ_{\boldsymbol{w}}(x) - Q_{\boldsymbol{w}}(x)\right)\end{aligned}$$

If we look at the average change this induces under stationary distribution $\mu(x)$ we get

$$\begin{aligned}\mathbb{E}_\mu[\boldsymbol{g}(\boldsymbol{w})] &= \sum_x \boldsymbol{\phi}(x)\mu(x)\left(SQ_w(x) - Q_w(x)\right)\\ &= \Phi^\top MS\Phi\boldsymbol{w} - \Phi^\top M\Phi\boldsymbol{w}\end{aligned} \tag{6.7}$$

where we use the definition that $\boldsymbol{Q}_{\boldsymbol{w}} = \Phi\boldsymbol{w}$. So the change in the algorithm is stationary when

$$0 = \mathbb{E}_\mu\left[\boldsymbol{g}\left(\boldsymbol{w}^*\right)\right] = \Phi^\top MS\Phi\boldsymbol{w}^* - \Phi^\top M\Phi\boldsymbol{w}^*$$

Rearranging and applying Φ to both sides gives

$$\begin{aligned}\Phi\boldsymbol{w}^* &= \Phi\left(\Phi^\top M\Phi\right)^{-1}\Phi^\top MS\Phi\boldsymbol{w}^*\\ &= \Pi S\Phi\boldsymbol{w}^*.\end{aligned}$$

the last part follows by the definition of the projection Π (6.5).

Differential Equation Analysis. The analysis here is essentially the same as the differential equation analysis in Section 6.1. Again the change in $\boldsymbol{w}(t)$ is approximated by the ODE

$$\frac{d\boldsymbol{w}(t)}{dt} = -A(\boldsymbol{w}(t))$$

where now

$$A(\boldsymbol{w}(t)) = \mathbb{E}_\mu\left[\boldsymbol{g}(\boldsymbol{w}(t))\right] = \Phi^\top M(I-S)\Phi(\boldsymbol{w}-\boldsymbol{w}^*)\,.$$

Note that A is no longer a linear function in $\boldsymbol{w}(t)$. 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 5a). Thus

$$\frac{dL}{dt} = \nabla L(\boldsymbol{w})\frac{dw}{dt} \leq -(\boldsymbol{w}-\boldsymbol{w}^*)^\top A(\boldsymbol{w}-\boldsymbol{w}^*).$$

This implied $L(\boldsymbol{w}(t)) \to 0$ and from this we can argue that $\boldsymbol{w}(t) \to \boldsymbol{w}^*$.

Approximation Error. If we let $\tau(\boldsymbol{w}^*)$ be the stopping rule induced by $\boldsymbol{w}^*$ from the above convergence argument then we can argue that its reward behaves similarly to projecting the optimal policy:

$$\mathbb{E}\left[V(x_0)\right] - \mathbb{E}\left[R_{\tau(\boldsymbol{w}^*)}(x_0)\right] \leqslant \frac{2\beta}{(1-\beta)^2}\left\|\Pi Q^* - Q^*\right\|_\mu$$

The argument is similar to the TD(0) case, previously. However, as discussed the argument is complecated by the fact the Q-factor for $\tau(\boldsymbol{w})$ is not the same as $Q(x;\boldsymbol{w})$. The full argument is proven more formally in (5) below.

Some Formal Analysis. Here we present the main ingredients applied in this section. (Excluding the differential equation argument.)

Proposition 5.
a) The maps S, ΠS and $S_{\boldsymbol{w}}$ are β-contractions with respect to the norm $\|\cdot\|_\mu$, that is

1. $\left\|SQ - SQ'\right\|_\mu \leqslant \beta\left\|Q-Q'\right\|_\mu$.
2. $\left\|\Pi SQ - \Pi SQ'\right\|_\mu \leqslant \beta\left\|Q-Q'\right\|_\mu$.

3. $\|S_{\boldsymbol{w}}Q - S_{\boldsymbol{w}}Q'\|_\mu \leqslant \beta \|Q - Q'\|_\mu$.

b)

$$(\boldsymbol{w} - \boldsymbol{w}^*)\, \mathbb{E}_\mu[\boldsymbol{g}(\boldsymbol{w})] < 0 \qquad \forall \boldsymbol{w} \neq \boldsymbol{w}^*$$

and

$$\mathbb{E}_\mu\left[\boldsymbol{g}\left(\boldsymbol{w}^*\right)\right] = 0$$

where $\boldsymbol{w}^*$ solves the equation

$$\Pi(S\Phi\boldsymbol{w}^*) = \Phi\boldsymbol{w}^*$$

c)

$$\mathbb{E}\left[V\left(x_0\right)\right] - \mathbb{E}\left[R_{\tau(\boldsymbol{w}^*)}\left(x_0\right)\right] \leqslant \frac{2\beta}{(1-\beta)^2} \|\Pi Q^* - Q^*\|_\mu$$

where here it is assumed that the expection 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

$$\|\Pi SQ - \Pi SQ'\|_\mu \leqslant \|SQ - SQ'\|_\mu \ .$$

So it remains to show that S is a contraction, which holds as follows.

$$\begin{aligned}
\|SQ - SQ'\|_\mu &\leq \beta \|P \max\{\bar{r}, Q\} - P \max\{\bar{r}, Q'\}\|_\mu \\
&\leq \beta \|\max\{\bar{r}, Q\} - \max\{\bar{r}, Q'\}\|_\mu \\
&\leq \beta \|Q - Q'\|_\mu .
\end{aligned}$$

Here we use the fact that $\|PR\|_\mu \leq \|R\|_\mu$ which we showed in Proposition 4 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 $S_{\boldsymbol{w}}$. We let $F_{\boldsymbol{w}}$ be the future value given by

$$F_{\boldsymbol{w}}Q(x) := \begin{cases} \bar{r}(x) & \text{if } \bar{r}(x) \geq \Phi w(x), \\ Q(x) & \text{otherwise .} \end{cases}$$

So $S_{\boldsymbol{w}}Q = r + \beta F_{\boldsymbol{w}}Q$. Now

$$\|S_{\boldsymbol{w}}Q - S_{\boldsymbol{w}}Q'\|_\mu = \beta \|PF_{\boldsymbol{w}}Q - PF_{\boldsymbol{w}}Q'\|_\mu \leq \beta \|F_{\boldsymbol{w}}Q - F_{\boldsymbol{w}}Q'\|_\mu \leq \beta \|Q - Q'\|_\mu$$

In the first inequality we use that $\|P\|_\mu \leq 1$, from Proposition 4, and the final inequality holds because

$$\begin{aligned}
\|F_{\boldsymbol{w}}Q - F_{\boldsymbol{w}}Q'\|_\mu^2 &= \sum_x \mu(x) |Q(x) - Q'(x)|^2 \mathbb{I}[\bar{r}(x) < \Phi\boldsymbol{w}(x)] \\
&\leq \sum_x \mu(x) |Q(x) - Q'(x)|^2 = \|Q - Q'\|_\mu
\end{aligned}$$

b) The following argument relies on the fact that ΠS is a contraction:

$$\begin{aligned}
&(\boldsymbol{w}-\boldsymbol{w}^*)\,\mathbb{E}_\mu[\boldsymbol{g}(\boldsymbol{w})]\\
&=(\boldsymbol{w}-\boldsymbol{w}^*)\,[\Phi^\top M S\Phi\boldsymbol{w}-\Phi^\top M\Phi\boldsymbol{w}]\\
&=(\boldsymbol{w}-\boldsymbol{w}^*)\,\Phi^\top M[\Pi S\Phi\boldsymbol{w}-\Phi\boldsymbol{w}]\\
&=(\boldsymbol{w}-\boldsymbol{w}^*)\,\Phi^\top M\,[\Pi S\Phi\,(\boldsymbol{w}-\boldsymbol{w}^*)-\Phi\boldsymbol{w}+\Phi\boldsymbol{w}^*]\\
&=(\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*)^\top M\Pi S\,(\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*)-(\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*)^\top M\,(\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*)\\
&\leqslant \|\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*\|_\mu\,\|\Pi S\Phi\boldsymbol{w}-\Pi S\Phi\boldsymbol{w}^*\|_\mu-\|\Phi\boldsymbol{w}-\Phi\boldsymbol{w}^*\|_\mu^2\\
&\leqslant -(1-\beta)\|\Phi\boldsymbol{w}-\Phi\boldsymbol{w}\|_\mu^2
\end{aligned}$$

In the first equality, we apply (6.7). 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)

$$\begin{aligned}
&\beta\mathbb{E}\left[V\left(x_0\right)\right]-\beta\mathbb{E}\left[V^\tau_{\boldsymbol{w}^*}\left(x_0\right)\right]\\
&=\mathbb{E}\left[r\left(x_0\right)-\beta PV\left(x_0\right)\right]-\mathbb{E}\left[r\left(x_0\right)-\beta PV^\tau_{\boldsymbol{w}^*}\left(x_0\right)\right]\\
&=\left|\mathbb{E}\left[Q\left(x_0\right)-Q^\tau_{\boldsymbol{w}^*}\left(x_0\right)\right]\right|\\
&\leq \|Q-Q^\tau_{\boldsymbol{w}}\|_\mu
\end{aligned}$$

The last inequality, above, applies Jensen's Inequality. Thus

$$\mathbb{E}\left[V\left(x_0\right)\right]-\mathbb{E}\left[V^\tau_{\boldsymbol{w}^*}\left(x_0\right)\right]\leqslant\beta^{-1}\left\|Q-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu \tag{6.8}$$

Similar to the argument in Proposition 4 we have

$$\begin{aligned}
\left\|Q-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu&=\left\|Q-S\left(\Phi\boldsymbol{w}^*\right)+S^\tau\left(\Phi\boldsymbol{w}^*\right)-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu\\
&\leqslant\|SQ-S\left(\Phi\boldsymbol{w}^*\right)\|_\mu+\left\|S^\tau\left(\Phi\boldsymbol{w}^*\right)-S^\tau(Q^\tau_{\boldsymbol{w}^*})\right\|_\mu\\
&\leqslant\beta\left\|Q-\Phi\boldsymbol{w}^*\right\|_\mu+\beta\left\|\Phi\boldsymbol{w}^*-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu\\
&\leqslant 2\beta\left\|Q-\Phi\boldsymbol{w}^*\right\|_\mu+\beta\left\|Q-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu
\end{aligned}$$

Here we use that $S\left(\Phi\boldsymbol{w}^*\right)=\Phi\boldsymbol{w}^*=S^\tau\left(\Phi\boldsymbol{w}^*\right)$ and the contraction property for both S and S^τ.

Thus, rearranging the above gives,

$$\left\|Q-Q^\tau_{\boldsymbol{w}^*}\right\|_\mu\leqslant\frac{2\beta}{1-\beta}\left\|Q-\Phi\boldsymbol{w}^*\right\|_\mu$$

Applying (6.8) to this gives

$$\mathbb{E}\left[V\left(x_0\right)\right] - \mathbb{E}\left[V^{\tau}_{\boldsymbol{w}^*}\left(x_0\right)\right] \leqslant \frac{2}{(1-\beta)^2}\|Q - \pi(Q)\|_{\mu}\,.$$

□

References

This section is based on reading Tsitsiklis and Van Roy [42], but also see Bertsekas and Tsitsiklis [6]. Further approaches to optimal stopping are discussed in the book of Glasserman [16].

6.3 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, may other choices exists.

Note that the probability of states and actions $\boldsymbol{x} = (x_0, ..., x_T)$ and $\boldsymbol{a} = (a_0, ..., a_{T-1})$ is

$$\pi_\theta(\boldsymbol{x}_T, \boldsymbol{a}_T) := \prod_{t=0}^{T-1} p(x_{t+1}|x_t, a_t)\pi_\theta(a_t|x_t)$$

The expected reward under policy π_θ is

$$R(\theta) := \mathbb{E}_{\pi_\theta}\left[Q(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}})\right] = \sum_{\boldsymbol{x},\boldsymbol{a}} Q(\boldsymbol{x}, \boldsymbol{a})\pi_\theta(\boldsymbol{x}, \boldsymbol{a})$$

where

$$Q(\boldsymbol{x}, \boldsymbol{a}) := \sum_{t=0}^{T-1} \beta^t r(x_t, a_t)$$

Differentiating the Reward Objective. We now cover a number calctlations where we differentiate the reward objective over θ. These are summarized by the following theorem.

Theorem 10 (The Policy Gradient Theorem)**.**

$$\begin{aligned}\nabla_\theta R(\theta) &= \mathbb{E}_{\pi_\theta}\Big[Q(\boldsymbol{x},\boldsymbol{a})\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a})\Big] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \beta^t Q_t \log \pi_\theta(a_t|x_t)\right]\end{aligned}$$

where $Q_t = \sum_{s=t}^{T-1} \beta^{s-t} r(x_s, a_s)$.

Proof. We can differentiate the reward

$$\begin{aligned}\nabla_\theta R(\theta) &= \sum_{\boldsymbol{x},\boldsymbol{a}} Q(\boldsymbol{x},\boldsymbol{a})\nabla_\theta \pi_\theta(\boldsymbol{x},\boldsymbol{a}) \\ &= \sum_{\boldsymbol{x},\boldsymbol{a}} \left\{Q(\boldsymbol{x},\boldsymbol{a})\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a})\right\} \pi_\theta(\boldsymbol{x},\boldsymbol{a}) \\ &= \mathbb{E}_{\pi_\theta}\left[Q(\hat{\boldsymbol{x}},\hat{\boldsymbol{a}})\nabla_\theta \log \pi_\theta(\hat{\boldsymbol{x}},\hat{\boldsymbol{a}})\right]. \end{aligned} \tag{6.9}$$

Thus we can apply stochastic gradient descent on the above objective by sampling

$$Q(x,a)\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a}).$$

Further we note that we can simply the derivative above,

$$\begin{aligned}\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a}) &= \nabla_\theta \left\{\log \prod_{t=0}^{T-1} p(x_{t+1}|x_t,a_t) + \log \prod_{t=0}^{T-1} \pi_\theta(a_t|x_t)\right\} \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|x_t)\end{aligned}$$

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

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(\hat{a}|x)] = \sum_a \pi_\theta(a|x)\frac{\nabla_\theta \pi(a|a)}{\pi_\theta(a|x)} = \nabla_\theta \underbrace{\left\{\sum_a \pi(a|x)\right\}}_{=1} = 0, \tag{6.10}$$

and, consequently for $s < t$,

$$\mathbb{E}_{\pi_\theta}\left[r(x_s,a_s)\log \pi_\theta(a_t|x_t)\right] = \mathbb{E}_{\pi_\theta}\left[r(x_s,a_s)\mathbb{E}_{\pi_\theta}\left[\log \pi_\theta(a_t|x_t)|x_t\right]\right] = 0 \tag{6.11}$$

Finally, we note that we can rearrange the objective (6.9) to account for the contribution from each individual state and action (x_t, a_t):

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[Q(x,a)\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a})\right] &= \mathbb{E}_{\pi_\theta}\left[\left(\sum_{s=0}^{T-1}\beta^s r(x_s,a_s)\right)\left(\sum_{t=0}^{T-1}\log \pi_\theta(a_t|x_t)\right)\right] \\
&= \sum_{t=0}^{T-1}\sum_{s=0}^{T-1}\beta^s \mathbb{E}_{\pi_\theta}\left[r(x_s,a_s)\log \pi_\theta(a_t|x_t)\right] \\
&\overset{\text{by (6.11)}}{=} \sum_{t=0}^{T-1}\sum_{s=t}^{T-1}\beta^s \mathbb{E}_{\pi_\theta}\left[r(x_s,a_s)\log \pi_\theta(a_t|x_t)\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1}\beta_t Q_t \log \pi_\theta(a_t|x_t)\right]
\end{aligned}$$

as required. □

Further for later use it is worth noting that in the above proof we also proved the following Lemma.

Lemma 13.

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(\hat{a}|x)] = 0\,.$$

Here the expectation is taken over $\hat{a}$ not x.

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

$$\theta \leftarrow \theta + \gamma\tilde{Q}\nabla_\theta \log \pi_\theta(\boldsymbol{x},\boldsymbol{a})$$

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 13, 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

$$\nabla_\theta R(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \beta^t \left(Q_t - B(x_t)\right) \log \pi_\theta(a_t|x_t)\right]$$

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

$$w \leftarrow w + \alpha \left(r + \beta V_w(\hat{x}) - V_w(x)\right) \nabla_w V_w(x) \tag{6.12a}$$

and we update the policy weights

$$\theta \leftarrow \theta + \alpha \left(Q_t - V_w(x_t)\right) \nabla_\theta \log \pi_\theta(\hat{a}|x) \tag{6.12b}$$

We implement these updates per step.

Actor-Critic. Notice when we apply the updates above (6.12) 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

$$\delta \leftarrow \left(r + \beta V_w(\hat{x}) - V_w(x)\right) \tag{6.13a}$$
$$w \leftarrow w + \alpha\delta \nabla_w V_w(x) \tag{6.13b}$$
$$\theta \leftarrow \theta + \alpha\delta \nabla_\theta \log \pi_\theta(\hat{a}|x) \tag{6.13c}$$

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 (6.13a), n-step TD would use:

$$\delta \leftarrow r_1 + \beta r_2 + \ldots + \beta^n r_n + \beta^{n+1} V_w(\hat{x}_{n+1}) - V_w(x)\,. \tag{6.13a$'$}$$

Chapter 7

Reinforcement Learning with Neural Networks

7.1 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 adaptions 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 summaary, 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 [47], except the Martingale Central Limit Theorem. Here instead see Hall and Heyde [19] and, for the Functional Martingale Central Limit Theorem, see Whitt [45].

Probability Inequalities

Below, unless stated otherwise, A_n, $n = 1, 2, ..$ are an events. X is a RV with mean μ, variance σ 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 of X.

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty} A_n\right) \leq \sum_{n=1}^{\infty} \mathbb{P}(A_n) \qquad \text{(Union bound)}$$

$$\mathbb{P}(X \geq x) \leq \frac{\mu}{x} \qquad \text{for } X \geq 0 \qquad \text{(Markov's inequality)}$$

$$\mathbb{P}(|X - \mu| \geq x) \leq \frac{\sigma^2}{x^2} \qquad \text{(Chebychev inequality)}$$

$$\mathbb{P}(X \geq x) \leq \exp\Big\{ - \max_{\theta \geq 0} \Big(\theta x - \log M_X(\theta)\Big)\Big\} \qquad \text{(Chernoff Bound.)}$$

$$\mathbb{P}(S_n \geq x) \leq \exp\Big\{ - \frac{x^2}{2nc^2}\Big\} \qquad \text{(Hoeffding's Inequality)}$$

$$X > 0 \implies \mathbb{E}[X] > 0 \qquad \text{(Positivity)}$$

$$f(\mathbb{E}[X]) \le \mathbb{E}[f(X)] \quad \text{for} \quad f \text{ convex} \qquad \text{(Jensen's Inequality)}$$

$$\|X\|_p \le \|X\|_q \quad \text{for } p \le q \qquad \text{(Minkowski's Inequality)}$$

$$\mathbb{E}[XY] \le \|X\|_2 \|Y\|_2 \qquad \text{(Cauchy-Schwartz)}$$

$$\mathbb{E}[XY] \le \|X\|_p \|Y\|_q \quad \text{for } \frac{1}{p} + \frac{1}{q} = 1 \qquad \text{(Holder's Inequality)}$$

Probability Limits

Here we assume $A_1, A_2, \ldots$ 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 $X_1, X_2, \ldots$ is a sequence of random variables (possibly not independent)

$$\mathbb{E}[X_n] \nearrow \mathbb{E}[X_\infty], \quad \text{for} \quad X_n \le X_{n+1}\,.$$

(Monotone Convergence Theorem)

$$\mathbb{E}\left[\liminf_{n\to\infty} X_n\right] \le \liminf_{n\to\infty} \mathbb{E}[X_n], \quad \text{for } X_n \ge 0. \qquad \text{(Fatou's Lemma)}$$

$$\mathbb{E}[X_n] \to \mathbb{E}[X_\infty] \quad \text{for } |X_n| \le Y \text{ with } \mathbb{E}Y < \infty$$

(Bounded Convergence Theorem)

Conditional Expectation

In what follows, we assume that X and Y are random variables which need not by real valued, e.g. $X = (X_1, X_2, \ldots, X_n)$ or $Y = (Y_t : t \in \mathbb{Z}_+)$, and Z is a real valued random variable.

Formally the conditional expecation 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

$$\mathbb{E}[g(X)f(X)] = \mathbb{E}[Zf(X)] \quad \text{for all } f \qquad \text{(Def. } \mathbb{E}[Z|X]\text{)}$$

Eg. take $g(x) = \mathbb{E}[Z|X = x]$.

$\mathbb{E}[\mathbb{E}[Z|X]f(X)] = \mathbb{E}[Zf(X)]$
$\mathbb{E}[Z_1Z_2|X] = Z_1\mathbb{E}[Z_2|X]$ if $Z_1 = f(X)$ (Taking out what is known)
$\mathbb{E}[\mathbb{E}[Z|X]|Y] = \mathbb{E}[Z|Y]$ if $Y = f(X)$ (Tower Property)
$\mathbb{E}[Z|X] = \mathbb{E}[Z]$ if Z, X are independent
$\mathbb{E}[Z|X, Y] = \mathbb{E}[Z|X]$ if Y is independent of X and Z.
(Role of independence)
$\mathbb{E}[g(X, Y)|X = x] = \mathbb{E}[g(x, Y)]$ if X and Y are independent.

Martingales and Stopping

We condition with respect to a sequence of random variables, in particular we let $F_n = (X_1, ..., X_n)$ (note F_n is a function of F_{n+1} so the Tower Property of the conditional expecation applies).

Suppose M_n is a sequence of RVs such that M_n is a function of F_n and $|M_n|$ has finite expectation then we say M_n is a Martingale if

$$\mathbb{E}[M_n|F_{n-1}] = M_{n-1} \qquad \text{(Martingale Property)}$$

Also for supermartingales and submartingales

$$\mathbb{E}[M_n|F_{n-1}] \leq M_{n-1} \qquad \text{(Supermartingale Property)}$$
$$\mathbb{E}[M_n|F_{n-1}] \geq M_{n-1} \qquad \text{(Submartingale Property)}$$

(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 :

$$M_\infty := \lim_{n\to\infty} M_n \quad \text{exists} \qquad \text{(Doob's Mg Convergence Thrm)}$$

• If M_t is a positive sub-martingale then

$$\mathbb{P}\Big(\sup_{n\leq t} M_n \geq x\Big) \leq \frac{\mathbb{E}[M_t]}{x} \qquad \text{(Doob's Sub-Mg Inequality)}$$

(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

$$\mathbb{P}\Big(\sup_{n\le t} M_n \ge x\Big) \le \exp\Big\{-\frac{x^2}{2tc}\Big\}. \qquad \text{(Azuma-Hoeffding)}$$

• 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

$$Y_t = Y_0 + M_t + Z_t \qquad \text{(Doob-Meyer Decomposition)}$$

moreover this decomposition is unique, with probability 1. Moreover, is 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 $p^{-1} + q^{-1} = 1$

$$\Big\|\sup_t M_t'\Big\|_p \le q \sup_t \|M_t'\|_p = q\|M_\infty'\|_p \qquad \text{(Doob's } \mathcal{L}^p \text{ inequality)}$$

• If M_t is a super-Mg with $\sup_t \mathbb{E}[|M_t|^p] < \infty$ then the limit :

$$X_n \to X_\infty, \quad \text{w.p. 1 and in } \mathcal{L}^p \quad \text{(Doob's } \mathcal{L}^p \text{ Mg Convergence Thrm)}$$

• For M_t a Mg with $\mathbb{E}M_t^2 < \infty$, $\forall t$

$$\frac{M_t}{\langle M\rangle_t} \to 0 \quad \text{on the event } \{\langle M\rangle_\infty = \infty\} \quad \text{(Strong Law for Martingales)}$$

where $\langle M\rangle_t := \sum_{n=1}^t \mathbb{E}[(M_n - M_{n-1})^2|F_{n-1}]$.
• Assume M_t is a Mg with bounded increments[1]

$$\frac{M_t}{\sqrt{\langle M\rangle_t}} \Rightarrow \mathcal{N}(0,1) \qquad \text{(Mg CLT)}$$

• Assume M_t is a Mg with bounded increments

$$\left(\frac{M_{nt}}{\sqrt{\langle M\rangle_{nt}}} : t \in [0,1]\right) \Rightarrow (B_t : t \in [0,1]) \qquad \text{(Mg CLT - v1)}$$

[1]This condition can be weakened.

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

$$\left(\frac{M_t^n}{\sqrt{\langle M^n \rangle_t}} : t \in [0,1] \right) \Rightarrow (B_t : t \in [0,1]) \qquad \text{(Mg CLT - v2)}$$

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 $(X_1, ..., X_t)$ is sufficient to know if T has happened yet or not.

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 118 (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,*

$$\sum_{t\in\{0,\delta,..,T\}} (B_{t+\delta} - B_t) = B_T \quad \textit{and} \quad \sum_{t\in\{0,\delta,..,T\}} (B_{t+\delta} - B_t)^2 \approx T$$

Ans 118. *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 119 (Continued)**.** *Discuss why it is reasonable to expect that*

$$\sum_{t\in\{0,\delta,..,T\}} \sigma(X_t)\,(B_{t+\delta} - B_t) \approx \int_0^T \sigma(X_t) dB_t$$

and

$$\sum_{t\in\{0,\delta,..,T\}} \mu(X_t)\,(B_{t+\delta} - B_t)^2 \approx \int_0^T \mu(X_t) dt.$$

Ans 119. *The first sum is approximation from a Riemann-Stieltjes integral, i.e.*

$$\int_0^T f(t) dg(t) \approx \sum_{t\in\{0,\delta,..,T\}} f(t)(g(t+\delta) - g(t)).$$

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 $(B_{t+\delta} - B_t)^2 \approx \delta$ *[118].*

Ex 120 (Continued). *If we inductively define X_t by the recursion*

$$X_{t+\delta} - X_t = \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta, \qquad t = 0, \delta, 2\delta, ...$$

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 120. *Sum to gain $X_T - X_0$ and apply approximations from [119].*

Ex 121 (Continued). *Let f be a twice differentiable function, argue that*

$$f(X_{t+\delta}) - f(X_t) \approx \left[f'(X_t)\mu(X_t) + \frac{\sigma(X_t)^2}{2}f''(X_t)\right]\delta + f'(X_t)\sigma(X_t)\left(B_{t+\delta} - B_t\right).$$

Ans 121. *Apply a Taylor approximation*

$$\begin{aligned}
&f(X_{t+\delta}) - f(X_t)\\
=&f(X_t + \sigma(X_t)(B_{t+\delta} - B_t) + \mu(X_t)\delta) - f(X_t)\\
=&f'(X_t)\left\{\mu\delta + \sigma\cdot(B_{t+\delta} - B_t)\right\} + \frac{f''(X_t)}{2}\left\{\mu\delta + \sigma\cdot(B_{t+\delta} - B_t)\right\}^2 + o(\delta)\\
=&f'(X_t)\left\{\mu\delta + \sigma\cdot(B_{t+\delta} - B_t)\right\} + \frac{f''(X_t)}{2}\sigma^2\cdot(B_{t+\delta} - B_t)^2 + o(\delta)
\end{aligned}$$

In the last equality we use that $(B_{t+\delta} - B_t) = o(\delta^{1/2})$, cf. [118].

Ex 122 (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 122. *Apply an interpolating sum to [121] and then apply [119].*

A.3 Gronwall's Lemma

We introduce a useful integration inequality due to Bellman.

Thrm 123 (Gronwall's Lemma)**.** *If $f : \mathbb{R}_+ \to \mathbb{R}_+$ is bounded above on each closed interval $[0, T]$ and satisfies*

$$f(T) \leq a(T) + \int_0^T b(t)f(t)dt \tag{A.1}$$

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

$$\begin{aligned}
\frac{\mathrm{d}v(t)}{\mathrm{d}t} &= -b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds + b(t)f(t)e^{-\int_0^t b(s)ds} \\
&\leq -b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds + a(t)b(t)e^{-\int_0^t b(s)ds} \\
&\quad + b(t)e^{-\int_0^t b(s)ds} \int_0^t b(s)f(s)ds \\
&= a(t)b(t)e^{-\int_0^t b(s)ds}.
\end{aligned}$$

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 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 124 (Utility Function)**.** *For $\mathcal{X} \subset \mathbb{R}^d$, a utility function is a function $U : \mathcal{X} \to \mathbb{R}$ 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 125 (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 126. *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 127. *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 127. *Define $\phi : \mathbb{R} \to \mathbb{R}$ 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 128 (Coefficient of Relative Risk Aversion)**.** *For a utility function $U : \mathbb{R} \to \mathbb{R}$ (twice differentiable) the Coefficient of Relative Risk Aversion is*

$$-x\frac{U''(x)}{U'(x)}.$$

Ex 129. *You have utility function U. You are offered a bet that increases you wealth w multiplicatively by $(1 + X)$ here X is a "small" positive is a 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 you mean is large but a large variance makes this less likely, and the coefficient of relative risk aversion decides the threshold.

Ans 129. *Accept if*

$$0 \leq \mathbb{E}[U(w(1+X)) - U(w)] \overset{Taylor}{\approx} \mathbb{E}\left[U'(w)wX + \frac{1}{2}U''(w)w^2X^2\right].$$

Def 130 (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 131. *A utility function is Iso-elastic if it is unchanged under multiplication: for all $c > 0$,*

$$\mathbb{E}U(X) \geq \mathbb{E}U(Y) \qquad \textit{iff} \qquad \mathbb{E}U(cX) \geq \mathbb{E}U(cY).$$

I.e. the utility only cares about the relative magnitude of the risk.

Ex 132. *Show that a utility function is iso-elastic iff it is a CRRA utility (up-to an affine transform).*

Ans 132. *By [127], its immediate that CRRA implies isolastic. Further by [127], $\forall c$, $U(cx) = a_c U(x) + b_c$ 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.

Index

Bibliography

[1] 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.

[2] Leemon C Baird III. Advantage updating. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

[3] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

[4] Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

[5] 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.

[6] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[7] David Blackwell. Discounted Dynamic Programming. *Ann. Math. Statist.*, 36(1):226–235, 1965.

[8] 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.

[9] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[10] P Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Texts in Applied Mathematics. Springer New York, 2013.

[11] Richard S Bucy and Peter D Joseph. *Filtering for stochastic processes with applications to guidance*. American Mathematical Soc., 1968.

[12] Y S Chow, H Robbins, and D Siegmund. *Great expectations: the theory of optimal stopping*. Houghton Mifflin, 1971.

[13] Mark HA Davis. Martingale methods in stochastic control. In *Stochastic Control Theory and Stochastic Differential Systems*, pages 85–117. Springer, 1979.

[14] J L Doob. *Classical Potential Theory and Its Probabilistic Counterpart: Advanced Problems*. Grundlehren der mathematischen Wissenschaften. Springer New York, 1984.

[15] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[16] Paul Glasserman. *Monte Carlo method in financial engineering*. 2004.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] M S Grewal and A P Andrews. *Kalman filtering: theory and practice with MATLAB, 4th*. Wiley, 2014.

[19] P Hall, C C Heyde, Z W Birnbaum, and E Lukacs. *Martingale Limit Theory and Its Application*. Communication and Behavior. Elsevier Science, 2014.

[20] Hado V Hasselt. Double q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[21] R Howard. Dynamic programming and Markov processes. 1960.

[22] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, 1994.

[23] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[24] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

[25] Harold Joseph Kushner and Dean S Clark. *Stochastic approximation methods for constrained and unconstrained systems*, volume 26. Springer Science & Business Media, 1978.

[26] Joseph LaSalle. Some extensions of liapunov's second method. *IRE Transactions on circuit theory*, 7(4):520–527, 1960.

[27] D A Levin, Y Peres, and E L Wilmer. *Markov Chains and Mixing Times*. American Mathematical Soc.

[28] Ljung. Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575, 1977.

[29] Aleksandr Mikhailovich Lyapunov. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992.

[30] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[31] J.R. Norris and J.R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[32] M L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[33] H Robbins and S Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22:400–407, 1951.

[34] 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.

[35] A N Shiryaev. *Optimal Stopping Rules*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg, 2009.

[36] Rayadurgam Srikant. *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.

[37] Ralph E Strauch. Negative Dynamic Programming. *Ann. Math. Statist.*, 37(4):871–890, 1966.

[38] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 1988.

[39] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. 2018.

[40] John N Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3):185–202, sep 1994.

[41] John N. Tsitsiklis and Benjamin Van Roy. Analysis of temporal-difference learning with function approximation. In *Advances in Neural Information Processing Systems*, 1997.

[42] 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.

[43] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[44] 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.

[45] Ward Whitt. Proofs of the martingale FCLT. *Probability Surveys*, 2007.

[46] Peter Whittle. *Optimization over time*. John Wiley & Sons, Inc., 1982.

[47] D. Williams. *Probability with Martingales*. Cambridge mathematical textbooks. Cambridge University Press, 1991.

[48] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML*, pages 928–936, 2003.