Unit 4: Financial Decision Approaches

Jeffrey D. Varner

Smith School of Chemical and Biomolecular Engineering
Cornell University, Ithaca NY 14853

Contents

List of Figures

List of Algorithms

1 Introduction

Decision making in random systems, particularly in financial contexts, involves navigating environments where outcomes are influenced by both predictable and unpredictable factors. Random systems are characterized by their reliance on probability distributions to describe the behavior of various elements. In finance, this could involve modeling the price movements of assets, interest rates, or other economic indicators. The goal is to understand and predict these movements to make informed decisions. Techniques such as Monte Carlo simulations, which use random sampling to estimate the probability of different outcomes, which we have already discussed, are commonly employed to assess the impact of uncertainty on investment strategies. In this unit, we will explore the use of Markov decision processes (MDPs) to model and solve decision-making problems in financial contexts. MDPs are a powerful tool for modeling sequential decision-making problems where outcomes are influenced by both random and controlled factors. However, MDPs require a detailed understanding of the underlying system dynamics and the ability to model these dynamics accurately. Thus, we will also explore model-free approaches, such as reinforcement learning, which learn by interacting with the environment and do not require explicit knowledge of the system dynamics. Let's start by exploring the fundamental concepts of Markov chains and Markov decision processes in Section 2. We'll then move on to model-free approaches such as Bandit problems in Section 3 and reinforcement learning in Section 4.

2 Markov Decision Processes

Markov chains, Markov models, and Markov decision processes are bedrock concepts in the field of stochastic processes, which are essential for understanding a wide range of applications in mathematics, economics, computer science, and beyond. These concepts provide a framework for modeling systems that undergo transitions from one state to another in a probabilistic manner. Let's start by looking Markov chains, and work our way up to Markov decision processes.

2.1 Markov Chains and Markov Models

A Markov chain is a type of stochastic process where the future state of the system depends only on the present state and not on the sequence of events that preceded it. This property is known as the Markov property. Markov chains are characterized by a set of states and transition probabilities between these states, which can be represented using a transition matrix. They are used to model various phenomena such as population dynamics, queueing systems, and random walks (which is the application we are most interested in).

Discrete-Time Markov Chains A discrete-time Markov chain is a sequence of random variables $X_1, \ldots, X_n$ with the Markov property, i.e., the probability of moving to the next state depends only on the present and not past states:

$$P(X_{n+1} = x | X_1 = x_1, \ldots, X_n = x_n) = P(X_{n+1} = x | X_n = y) \tag{1}$$

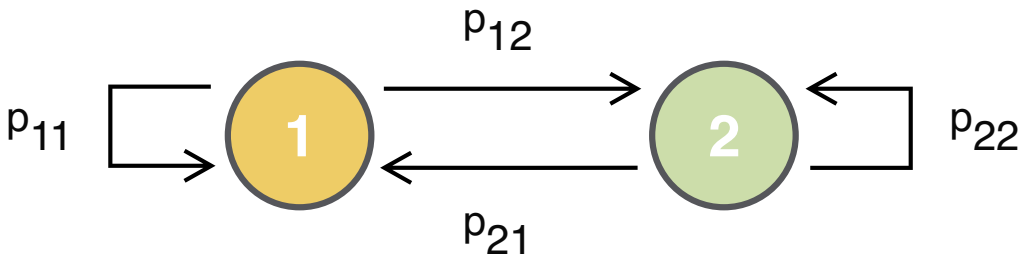

Fig. 1: Schematic of a discrete two-state time-invariant Markov model; p_{ij} denotes the time-invariant transition probability between state i and j.

For finite state spaces $\mathcal{S}$, the probability of moving from the state(s) $i \rightarrow j$ in the next turn, is encoded in the $|\mathcal{S}| \times |\mathcal{S}|$ transition matrix $p_{ij} \in \mathbf{P}$:

$$p_{ij} = P(X_{n+1} \ = \ j \mid X_n \ = \ i) \tag{2}$$

where $p_{ij} \geq 0$ for all $i, j \in \mathcal{S}$. The transition matrix $\mathbf{P}$ has interesting properties. First, the rows of transition matrix $\mathbf{P}$ sum to unity, i.e., each row encodes the probability of all possible outcomes in the next turn. Thus, it must sum to one. Second, if the transition matrix $\mathbf{P}$ is invariant, p_{ij} doesn't change as $n \rightarrow n+1 \ \forall n$. Finally, for a non-periodic Markov chain with a finite state space $\mathcal{S}$ and an invariant state transition matrix $\mathbf{P}$, the state vector at time j, denoted by π_j, has the property:

$$\sum_{s \in \mathcal{S}} \pi_{sj} = 1 \qquad \forall j \tag{3}$$

where $\pi_{sj} \geq 0, \forall s \in \mathcal{S}$. The state of the Markov chain at time step $n+1$, denoted by π_{n+1}, is given by:

$$\pi_{n+1} = \pi_1 \cdot (\mathbf{P})^n \tag{4}$$

where π_1 is the initial state vector, and $(\mathbf{P})^n$ is the transition matrix raised to the n-th power. Finally, for a non-periodic chain, a unique stationary distribution $\bar{\pi}$ exists; in the limit of large k the $\mathbf{P}^k$ converges to a rank-one matrix in which each row is the stationary distribution $\bar{\pi}$:

$$\lim_{k \rightarrow \infty} \mathbf{P}^k = \mathbf{1} \otimes \bar{\pi} \tag{5}$$

where $\mathbf{1}$ is a column vector of all 1s and $\otimes$ denotes the outer product.

Hidden Markov Models. Markov models extend the concept of Markov chains by incorporating additional complexities such as potentially continuous state spaces, time parameters or observable versus hidden states. These models are widely used in fields like economics and finance to predict future states of a system based on current observations. They can also be employed in machine learning for tasks like sequence prediction and natural language processing.

One common example of a Markov model (which we will use) is the hidden Markov model (HMM). A Hidden Markov Model (HMM) is a statistical model used to represent systems where the underlying processes are not directly observable, but can be inferred through observable sequences. The model is based on the concept of a latent Markov process, where the hidden states

influence the observable outputs. In HMMs, the relationship between the hidden states and observations is defined through two sets of probabilities: transition probabilities, which describe the likelihood of moving from one hidden state to another, and emission probabilities, which describe the likelihood of an observation given a hidden state (see Definition 1).

Definition 1: Hidden Markov Model

A hidden Markov model is defined by the tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathbf{P}, \mathbf{E})$.

- The state space $\mathcal{S}$ is the set of all possible *hidden* states s that the model $\mathcal{M}$ contains
- The observation space $\mathcal{O}$ is the set of all possible *observable* states o that the model $\mathcal{M}$ can emit
- The transition matrix $\mathbf{P}$ encodes the probability of moving from one hidden state to another hidden state at the next time step where $p_{ij} = P(X_{n+1} = j | X_n = i)$ must satisfy:

$$\sum_{j \in \mathcal{S}} p_{ij} = 1 \qquad \forall i \in \mathcal{S} \tag{6}$$

- The emission matrix $\mathbf{E}$ encodes the probability of emitting an observable state $o \in \mathcal{O}$ from a hidden state $s \in \mathcal{S}$ where the elements $e_{ij} = P(Y = j \,| X = i)$ must satisfy:

$$\sum_{j \in \mathcal{O}} e_{ij} = 1 \qquad \forall i \in \mathcal{S} \tag{7}$$

Let's consider the following pseudocode implementation for a hidden Markov model simulation (Algorithm 1).

Algorithm 1 Hidden Markov Model

procedure MARKOV-SIMULATION($\mathbf{A}, \mathbf{E}, \bar{\pi}, n$)
  $\dim \mathcal{S}, \dim \mathcal{O} \leftarrow$ `size`$(\mathbf{E})$ ▷ Dimension of the emission matrix $\mathbf{E}$ using `size`
  for $s = 1$ to $\dim \mathcal{S}$ **do**
    $d_H(s) \leftarrow$ `Categorical`$(\mathbf{A}[s, \dots])$ ▷ Construct a categorical distribution d_H
    $d_O(s) \leftarrow$ `Categorical`$(\mathbf{E}[s, \dots])$ ▷ Construct a categorical distribution d_O
  end for
  $s \leftarrow$ `rand`$(\bar{\pi})$ ▷ Generate initial state s from $\bar{\pi}$ using `rand` function
  for $i = 1$ to n **do**
    $s' \leftarrow$ `rand` $\circ\, d_H(s)$ ▷ Generate next state s' by sampling d_H distribution
    $o \leftarrow$ `rand` $\circ\, d_O(s')$ ▷ Generate observation o by sampling d_O distribution
    $s \leftarrow s'$ ▷ Update state s to s'
  end for
end procedure

The stationary distribution $\bar{\pi}$ is a probability distribution over the state space $\mathcal{S}$, where each state $s \in \mathcal{S}$ occurs with probability $\bar{\pi}_s$. The stationary distribution $\bar{\pi}$ describes the steady-state

behavior of the Markov chain. We model the rows of $\mathbf{P}$, $\mathbf{E}$ and the stationary distribution $\bar{\pi}$ using Categorical distributions. Categorical distributions are discrete probability distributions modeling the probability of a random variable taking a particular outcome $k \in \mathcal{K}$:

$$P(X = k) = \pi_k \qquad \text{where} \qquad \sum_{k \in \mathcal{K}} \pi_k = 1 \tag{8}$$

Sampling from a categorical distribution constructed from the rows of $\mathbf{P}$ (or $\mathbf{E}$) will return the next state s' (or observation o).

2.2 Markov Decision Processes

A Markov decision process (MDP) provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision-maker. MDPs take their name from the mathematician Andrey Markov, as they are an extension of Markov chains, a stochastic model which describes a sequence of possible events in which the probability of each event depends only on the system state in the previous event.

At each time step, let's assume the system in some state s in a set of possible states $s \in \mathcal{S}$ and the decision maker may choose any action a from a set of possible actions $a \in \mathcal{A}$ that are available in state s. The system responds at the next time step by potentially moving into a new state s' and rewarding the decision maker $R_a(s, s')$. The probability that the system moves into a new state s' depends upon the chosen action a and the current state s; this probability is governed by a state transition function $P_a(s, s')$.

Definition 2: Markov Decision Process Tuple

A Markov decision process is the tuple $(\mathcal{S}, \mathcal{A}, R_a(s, s'), T_a(s, s'), \gamma)$ where:

- The state space $\mathcal{S}$ is the set of all possible states s that a system can exist in
- The action space $\mathcal{A}$ is the set of all possible actions a that are available to the agent, where $\mathcal{A}_s \subseteq \mathcal{A}$ is the subset of the action space $\mathcal{A}$ that is accessible from state s.
- An expected immediate reward $R_a(s, s')$ is received after transitioning from state $s \rightarrow s'$ due to action a.
- The transition $T_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$ denotes the probability that action a in state s at time t will result in state s' at time $t + 1$
- The quantity γ is a discount factor; the discount factor is used to weight the future expected utility.

Finally, a policy function π is the (potentially probabilistic) mapping from states $s \in \mathcal{S}$ to actions $a \in \mathcal{A}$ used by the agent to solve the decision task.

2.3 Policies and Value Functions

One immediate question that jumps out is what is a policy function π, and how do we find the best possible policy for our decision problem? To do this, we need a way to estimate how good (or bad) a particular policy is; the approach we use is called policy evaluation. Let's denote the expected

utility gained by executing some policy $\pi(s)$ from state s as $U^{\pi}(s)$. Then, an optimal policy function $\pi^{\star}$ is one that maximizes the expected utility:

$$\pi^{\star}(s) = \text{arg max } U^{\pi}(s) \tag{9}$$

for all $s \in \mathcal{S}$. We can iteratively compute the utility of a policy π. If the agent makes a single move, the utility will be the reward the agent receives by implementing policy π:

$$U_1^{\pi}(s) = R(s, \pi(s)) \tag{10}$$

However, if we let the agent perform two, three, or k possible iterations, we get a lookahead equation which relates the value of the utility at iteration k to $k+1$:

$$U_{k+1}^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s, \pi(s)) U_k^{\pi}(s') \tag{11}$$

As $k \to \infty$ the lookahead utility converges to a stationary value $U^{\pi}(s)$ (Definition 3).

Definition 3: Converged Policy Evaluation

Suppose we have a Markov decision process with the tuple $(\mathcal{S}, \mathcal{A}, R_a(s, s'), T_a(s, s'), \gamma)$. Then, the utility of the policy function π equals:

$$U^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s, \pi(s)) U^{\pi}(s') \tag{12}$$

The utility associated with an optimal policy $\pi^{\star}$ is called the optimal utility $U^{\star}$.

Definition 3 gives us a method to compute the utility for a particular policy $U^{\pi}(s)$. However, suppose we were given the utility and wanted to estimate the policy $\pi(s)$ from that utility. Given a utility U, we can estimate a policy $\pi(s)$ using the Q-function (action-value function):

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s, a) U(s') \tag{13}$$

For discrete state and action spaces, (13) gives a $|\mathcal{S}| \times |\mathcal{A}|$ array, where the utility is given by:

$$U(s) = \max_a Q(s, a) \tag{14}$$

and the policy $\pi(s)$ is:

$$\pi(s) = \text{arg} \max_a Q(s, a) \tag{15}$$

2.4 Summary

Fill me in

3 Bandit Problems

The multi-armed bandit problem is a fundamental problem in machine learning and decision-making. It refers to a scenario where an agent must repeatedly choose between multiple actions, each with an unknown reward distribution. Each action can be considered a slot machine, or a one-armed bandit with potentially different payout distributions. The goal of the agent is to maximize its total reward over time while learning the reward distributions of the different actions. This presents a tradeoff between exploitation, where the agent chooses the action with the highest estimated reward so far, and exploration, where it selects an action with an uncertain reward to learn more about its distribution. The multi-armed bandit problem has practical applications in various fields, such as clinical trials, online advertising, and recommender systems.

3.1 Binary Bernoulli Bandits

A bandit problem is a sequential game, played between an agent and the world, over T turns called the horizon. In each turn the agent chooses an action $a \in \mathcal{A}$, implements this action, and receives a reward r from the world. The goal of the agent is to maximize the total reward over the horizon, i.e., the agent learns the differnce between good and bad actions based on the rewards it receives. There are many different bandit problem varients, so let's focus on a particular sub-type of bandit problem, namely, the Bernoulli bandit problem (Definition 4).

Definition 4: Bernoulli Bandit Problem

A binary Bernoulli bandit problem is a multi-arm bandit problem where each action $a \in \mathcal{A}$ has a binary reward distribution governed by a Bernoulli random variable with unknown success parameter p_a:

- An agent chooses between K possible actions $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ where the success or failure of any action $a_i \in \mathcal{A}$ is governed by Bernoulli random variable with unknown success parameter p_{a_i}.
- During the game, the agent learns the success probabilities p_{a_i} by experimenting with the world. If action a is successful, the agent receives a reward $r = 1$, otherwise $r = 0$.
- The success probabilities are modeled by the agent as $p_a \sim \text{Beta}(\alpha, \beta)$, where the agent learns updates to the parameters (α, β) of the Beta distribution by experimenting with the world.

The agent's objective is to choose actions that maximize the cumulative reward over the game *horizon*.

3.2 Thompson Sampling

In decision-making, the multi-armed bandit problem serves as a classic framework for understanding the trade-off between exploration and exploitation. The challenge for the decision agent lies in balancing the exploration of new actions to gather information and the exploitation of known actions to maximize rewards. Thompson Sampling, which was first published in the context of

clincal trials in the 1930s [1], has emerged as a prominent approach to address this dilemma [2]. It operates by maintaining probability distributions (implemented as Beta distributions in our case) over the potential success probabilities of each action p_a and sequentially updates these distributions based on observed outcomes. By sampling from these distributions, Thompson Sampling probabilistically favors options that appear promising, thus efficiently navigating the exploration-exploitation trade-off. The algorithm is simple, intuitive, and has been shown to perform well in practice, and it has some theoretical guarantees of optimality under certain conditions [3]. A pseudocode implementation of ϵ-Greedy Thompson Sampling is provided in Algorithm 2.

Algorithm 2 ϵ-Greedy Thompson Sampling

procedure ϵ-GREEDY THOMPSON SAMPLING
    Initialize the α, β-vectors, the success $\mathbf{S}$ and fail $\mathbf{F}$ vectors and action set $\mathcal{A}$
    for $k \in 1$ to horizon **do** ▷ Play the game for horizon rounds
        $a \leftarrow \texttt{nothing}$
        if $\texttt{rand} < \epsilon$ **then** ▷ With threshold ϵ, we explore
            $a \leftarrow \texttt{uniform}(\mathcal{A})$ ▷ Exploration generates a uniform random $a \in \mathcal{A}$
        else ▷ With probability $1 - \epsilon$, we exploit
            $\mathbf{p} \leftarrow \{\text{Beta}(\alpha(k) + \mathbf{S}(k), \beta(k) + \mathbf{F}(k))\}_{k=1}^{\dim \mathcal{A}}$ ▷ Draw $\dim \mathcal{A}$ samples
            $a \leftarrow \arg\max_a \mathbf{p}$ ▷ Select the action with the highest success probability
        end if
        $r \leftarrow \texttt{world}(a)$ ▷ Observe the reward $r \in 0, 1$ from the `world` from action a
        $\mathbf{S}(a) \leftarrow \mathbf{S}(a) + r$ ▷ Update the success count for action a
        $\mathbf{F}(a) \leftarrow \mathbf{F}(a) + (1 - r)$ ▷ Update the fail count for action a
    end for
end procedure

3.3 Applications of Multi-Armed Bandits

4 Model Free Reinforcement Learning

4.1 Q-Learning in Discrete Spaces

4.2 Q-Learning in Continuous Spaces

4.3 Applications of Q-Learning

4.4 Summary

References

1. Thompson WR. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. Biometrika. 1933;25(3/4):285–294.
2. Russo D, Roy BV, Kazerouni A, Osband I, Wen Z. A Tutorial on Thompson Sampling; 2020. Available from: https://arxiv.org/abs/1707.02038.
3. Agrawal S, Goyal N. Analysis of Thompson Sampling for the multi-armed bandit problem; 2012. Available from: https://arxiv.org/abs/1111.1797.