January 15, 2020 Notes:

- Guided Tour of Chapter 1
- Before class, read chapter in book corresponding to lecture

- Markov Processes and Markov Reward Processes
  - ↳ This lecture is considered basic
    - ↳ Get comfortable with mathematical foundations and foundational code.
  - ↳ Markov Process is often referred to as a Markov Chain
  - ↳ Process: Time-sequenced random outcomes
  - ↳ State: Internal Representation S_t
  - ↳ We are interested in $P[S_{t+1} \mid S_t, S_{t-1}, \ldots, S_0]$

- Consider random walks of stock prices $X_t = S_t$

$$P[X_{t+1} = X_t + 1] + P[X_{t+1} = X_t - 1] = 1$$

- Example 1: Logistic Function

$$P[X_{t+1} = X_t + 1] = \frac{1}{1 + \exp[-\alpha_1 (L - X_t)]}$$

(L: Level = Long-term expectation)

  - Mean reverting process
  - This is a Markov Process: Future independent of past given present

- Markov Processes offer great computational benefit

- Modeling issue: State captures all relevant history
  - ↳ Don't want too much information to blow up space
  - ↳ State is sufficient history of future

- Example 2:

$$P[X_{t+1} = X_t + 1] = \begin{cases} 0.5\,(1 - \alpha_2 (X_t - X_{t-1})) & \text{if } t > 0 \\ 0.5 & \text{if } t = 0 \end{cases}$$

↳ To make this a Markov Process:

$$S_t = (X_t,\ X_t - X_{t-1})$$

- Example 3: Movement depends on entire history

Unit Sigmoid Curve

$$P[X_{t+1} = X_t + 1] = \begin{cases} \dfrac{1}{1 + \left(\frac{U_t}{D_t} - 1\right)^{\alpha_3}} & \text{if } t > 0 \\ 0.5 & \text{if } t = 0 \end{cases}$$

↳ Movement bias to reverse entire history

$$S_t = (U_t,\ D_t)$$

↳ X_t is not in state, but can be derived from:

$$X_t = X_0 + U_t - D_t$$

- Make another example
- Generate Simulation Paths with different Strength parameters
- Run multiple simulation traces and plot distributions of outcomes at multiple points of time
  - ↳ Plot variances as a function of α

- **Definition of Discrete Time, Countable States**
  - Subset of state space denoted terminal space ends Markov process upon occurance
  - Non-Stationary Transition probabilities have time-dependent transition probabilities.
  - In this course, if not specified, assume Stationary Markov Process
    - ↳ **Stationary Transition Probability Function**

$$P(s, s') = P[S_{t+1} = s' \mid S_t = s] \quad \text{for } s \in \mathcal{N},\ s' \in \mathcal{S}$$

  - A lot of authors describe termination with absorbing state, but we do not.
    - ↳ Does not work well with MDP
    - ↳ Programmatically challenging
  - In code we explicitly specify terminal states
  - In design of software and model, Consider:
    1. Transition Probabilities, P
    2. Probability distribution of start states, μ
  - **Episodic**: All traces return to a terminating state

↳ Some processes do not terminate and are called Continuing

- Code for MarkovProcess :

  - Abstract methods must be defined in derived classes
  - In code, specify types of each input and output

    ↳ Transition takes a State and returns a probability distribution of next State

    ↳ Are programming concepts like generics, types, generators prerequisite for this course ?

  - is_terminal checks if transition distribution is None

  - Generators create simulation trace in a lazy manner - only when somebody calls it.

    ↳ Creates iterable step-by-step

    ↳ Lets you deal with infinite States

  - Instead of representing P as a matrix of transition probabilities, use curried representation

    Mapping [S , Optional [FiniteDistribution [S]]

    S ↑ State

    FiniteDistribution [S] ↑ Look at distributions.py

  - Look at printed values and graphical views of transition probabilities

  - FiniteMarkovProcess is a concrete class of abstract class MarkovProcess

    ↳ Makes concrete by implementing Transition method

- **Inventory Example:** Inventory Control

α := On-hand Inventory
β := On-order Inventory
} Constitutes State

C := Store Capacity

↳ Observe State S_t: (α, β) at 6 pm store close

↳ Order quantity := $\max(C - (\alpha + \beta), 0)$

↳ Order takes 36 hours to ship

↳ Demand is Poisson, i

↳ Inventory Sold: $\max(\alpha + \beta, i)$

↳ Why is it max and not min?

$S := \{(\alpha, \beta) : 0 \leq \alpha + \beta \leq C\}$

... See slide 17 and review textbook to frame Markov Process

- A Markov Chain w/o terminal state yields State Frequency π
- For stationary process with finite states:

$\pi(s)$ is an eigenvector of P^T with eigenvalue of 1

- **Markov Reward Process** calculating expected accumulated reward, but does not optimize reward

$$P[(R_{t+1}, S_{t+1}) \mid S_t \ldots S_0] = P[(R_{t+1}, S_{t+1}) \mid S_t]$$

↳ MarkovRewardProcess in code is an abstract class

↳ transition_reward is new abstract method

↳ transition is implemented using transition_reward

* Course will focus on software design driven by math.

- Reward is often expressed as an expected reward (See slide 22)

↳ You can express reward function as a function of State alone

- You can extend inventory example with concept of reward

↳ There is a cost of "interest on inventory"

↳ There is also a cost associated with disappointed customers if inventory is not available ("soft money")

↳ Emotional factors for customers

$$P_R : N \times R \times S \longrightarrow [0,1]$$

$$\Updownarrow$$

$$N \longrightarrow (S \times R \longrightarrow [0,1])$$

- Define Return G_t

↳ Enables convergence (enable mathematical tractability)

↳ Interest Rates, inflation, TVM

- If all sequences terminate, set discount factor equal to 1

↳ E.g. $\gamma = 1$ for AI in car navigation

$$V(s) = \mathbb{E}[G_t \mid S_t = s]$$

↑ Value function

- Bellman Equation is a recursive equation for $V(s)$
  - ↳ See slide 27
    - ↳ Requires matrix inversion, which can be intractible. Use DP instead!

- intuition → picture → Math → Programming