**Markov Chains: A Powerful Tool for Modeling Sequential Data**

[[Arjun Prakash](https://medium.com/@arjunprakash027?source=post_page-----202ee8e00b72--------------------------------)](https://medium.com/@arjunprakash027?source=post_page-----202ee8e00b72--------------------------------)

[Arjun Prakash](https://medium.com/@arjunprakash027?source=post_page-----202ee8e00b72--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F376a20930ef1&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40arjunprakash027%2Fmarkov-chains-a-powerful-tool-for-modeling-sequential-data-202ee8e00b72&user=Arjun+Prakash&userId=376a20930ef1&source=post_page-376a20930ef1----202ee8e00b72---------------------post_header-----------)

5 min read

·

Oct 21

135

2

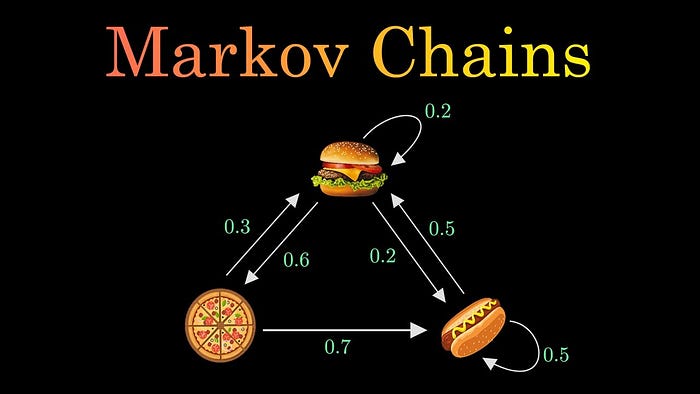


image from [here](https://www.google.com/url?sa=i&url=https%3A%2F%2Fm.youtube.com%2Fwatch%3Fv%3Di3AkTO9HLXo&psig=AOvVaw0G1mMLrpa7RBrsn2qdgRV_&ust=1697914282789000&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCODBoN-lhYIDFQAAAAAdAAAAABAE)

**What is Markov chain**

Markov chain is a type of mathematical model that describes a sequence of possible events that follow Markov property. Markov property states each event (about to happen) only depends on current state and not on previous states.

Let's take an example, let's consider a coin that is biased 60% for head and 40% for tail. The continuous flip of the coin is an Markov chain, each value (head or tail) depends only on current flip and not on previous flips.

**Representing a Markov Chain**

A discrete time Markov chain is a sequence of random variables X1, X2, X3, … with the Markov property, such that the probability of moving to the next state depends only on the present state and not on the previous states. Putting this is mathematical probabilistic formula:

*Pr( Xn+1 = x | X1 = x1, X2 = x2, …, Xn = xn) = Pr( Xn+1 = x | Xn = xn)*

as we can see probability of Xn+1 depends only on Xn that precedes it and **not any other previous states.**

The possible values of Xi forms a discrete set called as **state space**, and state space can be anything: numbers, words, weather anything you name.

The change from one state to another is called **state transition** and the probability associated with each transition is called **transition probability**.

If the Markov chain has N possible states, the matrix will be an N x N matrix, such that entry (I, J) is the probability of transitioning from state I to state J. Additionally, the transition matrix must be a stochastic matrix, a matrix whose entries in each row must add up to exactly 1, because each row is its own probability distribution.

**An Example**

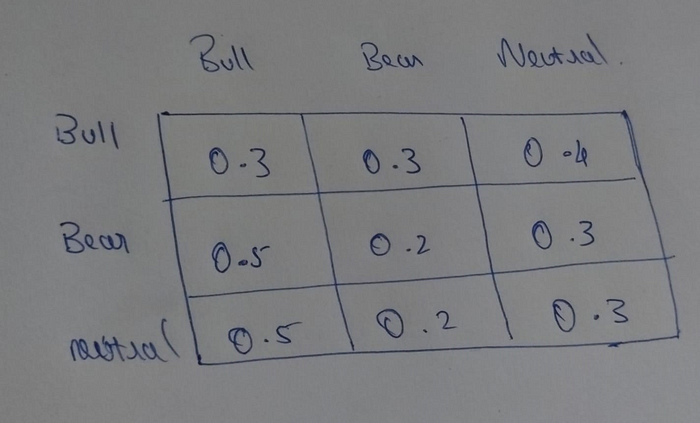
Lets take example of a imaginary stock market where we have a idea of what will happen next day if we know what's happening today.

If its a bull market today, there is 30% change it will be bull market tomorrow, 40% chance it will be a neutral market and 30% chance it will be a bear market (see all add up to 100% since its a probability distribution)

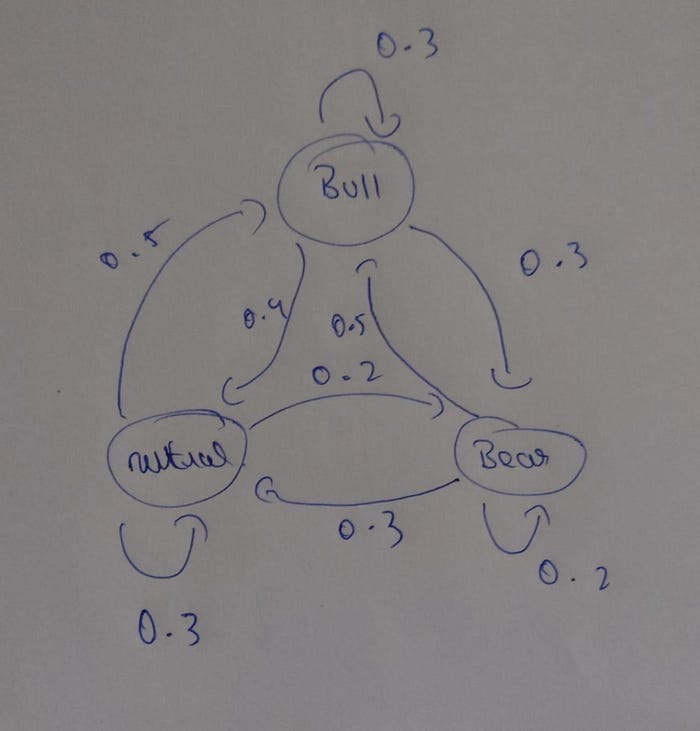
If its bear market today, a 20% chance it will be bear tomorrow, 50% it will be bull and 30% it will be neutral.

If its a neural market today, 50% chance it will be a bull market, 20% it will be bear market and 30% it will be a neutral market.

Now this is called state transition and to convert it into a state transition matrix, it will look like this:



and depicting it in form of directed graph:



**How to calculate the probability of next state**

Now lets calculate the probability of next state given current state using multiplication of matrix.

this is the possible states -> [bull,bear,neutral]

lets say today we start with a bear market

so, our current state is **[0,1,0]**

and our transition matrix is

**[[0.3,0.3,0.4],**

**[0.5,0.2,0.3],**

**[0.5,0.2,0.3]]**

on multiplication we get **[0.5,0.2,0.3]**

this states that the probability of going from bear to bull market is 0.5, bear to bear is 0.2 and bear to neutral is 0.3

What if I want to calculate probability for 3 days? Just do the same steps but with a little twist.

when we started, our current state was [0,1,0]…and we got an probability for next state as [0.5,0.2,0.3], now to calculate probability of 3rd day, we consider [0.5,0.2,0.3] as our current state and multiply it with transition matrix to get probability of 3rd day.

we get probability as [0.33,0.38,0.38]

to get probability of 4th day we take current state as [0.33,0.38,0.38] and multiply it with transition matrix and so on for n number of days.

**Code in Python**

import numpy as np  
  
class MarkovChain:  
  
 def \_\_init\_\_(self, transition\_matrix,states):  
 self.transition\_matrix = np.array(transition\_matrix)  
 self.states = states  
  
 def next\_state(self,current\_state:str) -> str:  
 current\_state\_matrix = np.array([0]\*len(self.states))  
 current\_state\_matrix[self.states.index(current\_state)] = 1  
 next\_state = current\_state\_matrix.dot(self.transition\_matrix)  
 random\_state\_probablity = np.random.choice(len(self.states),p=next\_state)  
 return self.states[random\_state\_probablity]  
   
 def nStates(self,starting\_state:str,nstates:int):  
 current\_state = np.array([0]\*len(self.states))  
 current\_state[self.states.index(starting\_state)] = 1  
 n\_states = []  
 state\_transition\_record = []  
 for i in range(nstates):  
 state\_transition\_record.append(current\_state)  
 next\_state = current\_state.dot(self.transition\_matrix)  
 random\_state\_probablity = np.random.choice(len(self.states),p=next\_state)  
 n\_states.append(self.states[random\_state\_probablity])  
 current\_state = next\_state  
 return n\_states,state\_transition\_record

**Explanation**

**\_\_init\_\_(self, transition\_matrix, states):**

* This is the constructor method for the MarkovChain class.
* It initializes the Markov chain with a transition matrix and a list of states.
* The transition\_matrix is a 2D NumPy array representing the transition probabilities between states. Each row represents the current state, and each column represents the next state.
* The states is a list of state names. The order of states in this list should correspond to the rows and columns of the transition matrix.

**next\_state(self, current\_state: str) -> str:**

* This method takes the current state as input and returns the next state based on the transition probabilities.
* It first creates a one-hot encoded vector called current\_state\_matrix, which represents the current state as a probability distribution with probability 1 for the current state and 0 for all other states.
* It calculates the next state by multiplying current\_state\_matrix with the transition\_matrix.
* The np.random.choice function is used to randomly choose the next state based on the probabilities in the next\_state vector.
* The selected state name is returned.

**nStates(self, starting\_state: str, nstates: int):**

* This method generates a sequence of states starting from the starting\_state for a total of nstates steps.
* It initializes the current\_state as a one-hot encoded vector for the starting state.
* It maintains two lists: n\_states to store the generated sequence of states and state\_transition\_record to record the state at each step.
* It enters a loop that iterates nstates times:
* Appends the current state to state\_transition\_record.
* Calculates the next state by multiplying current\_state with the transition\_matrix.
* Randomly selects the next state based on the transition probabilities.
* Appends the selected state to the n\_states list.
* Updates the current\_state for the next iteration.
* Finally, it returns both the generated sequence of states (n\_states) and the recorded state transitions (state\_transition\_record).

**Application of Markov chains**

* **PageRank algorithm**: Markov chains are used to rank web pages based on the number and quality of links pointing to them. The web is modeled as a directed graph, where pages are nodes and links are edges. The damping factor accounts for the possibility of a random jump to any page.
* **Stock market prediction**: Markov chains can be used to model the behavior of the stock market, which can switch between different states, such as bull, bear, or stagnant. The transition probabilities can be estimated from historical data and used to forecast future market conditions.
* **Text generation**: Markov chains can be used to generate random and meaningful text messages, such as those produced by bots on Reddit. The text is generated by using word-to-word probabilities learned from a large corpus of text data.
* **Election forecasting**: Markov chains can be used to predict the outcome of elections, based on the past voting patterns and trends. Bootstrap percentiles can be used to calculate confidence intervals for these predictions.

If this blog was informational, please consider giving me a follow on medium and connect with me on [LinkedIn](https://www.linkedin.com/in/arjunprakash027/)