Markov Chains

Table of Contents

[Transition Matrix 1](#_Toc120900629)

[Applications 2](#_Toc120900630)

[Recurrent and Transient States 3](#_Toc120900631)

[Irreducible Markov Chains 4](#_Toc120900632)

[Long-Run Probabilities 7](#_Toc120900633)

[Limiting Probability 8](#_Toc120900634)

[Gambler’s Ruin Problem 10](#_Toc120900635)

[Proof 11](#_Toc120900636)

[Acceptable and Unacceptable States 16](#_Toc120900637)

[Markov Decision Processes 17](#_Toc120900638)

Let be a random process. If , then the process is said to be in state at time . In a **Markov Chain**, we assume that whenever the process is in state , there is a fixed probability that it will next be in state .

Formally,

is only dependent on .

## Transition Matrix

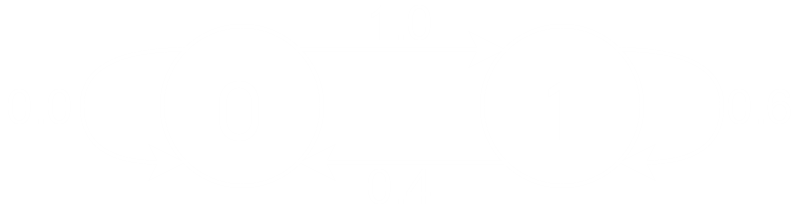
A transition matrix allows us to quickly visualize all possible values of . Suppose we need to decide whether it rains ) or not ), and we are given the following information:

1. If it rains today, it will rain tomorrow with .
2. If it does not rain today, it will rain tomorrow with .

Thus, the transition matrix looks like this:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | No  Rain | Rain |
|  | No  Rain | 0 | 1 |
| Rain | 0.4 | 0.6 |

It is also possible to have a state diagram that displays this information.



## Applications

Markov chains can be used with anything that has a sequence, such as:

* Coin Flips – Ideally, coin flips should be independent. However, real life conditions make it possible for the flips to be in a sequence.
* Stock Prices – It is possible to create a window of possible values based on the volatility of a stock.
* Reinforcement Learning

The transition matrix gives the probability of ending up in some state after a single transition. For example, the following transition matrix is for the probability of rain tomorrow depending on wither or not it rained today:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | No  Rain | Rain |
|  | No  Rain | 0 | 1 |
| Rain | 0.4 | 0.6 |

If we want to find the probability of ending up in some state after transitions, we have to get the th power of . For example, the following matrix gives the probability of rain 3 days from today.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | No  Rain | Rain |
|  | No  Rain | 0.24 | 0.76 |
|  | Rain | 0.304 | 0.696 |

Note that in the above matrix, the rows are for the state of today, i.e., if it rains today, it will rain 3 days from today with a probability of 0.696.

## Recurrent and Transient States

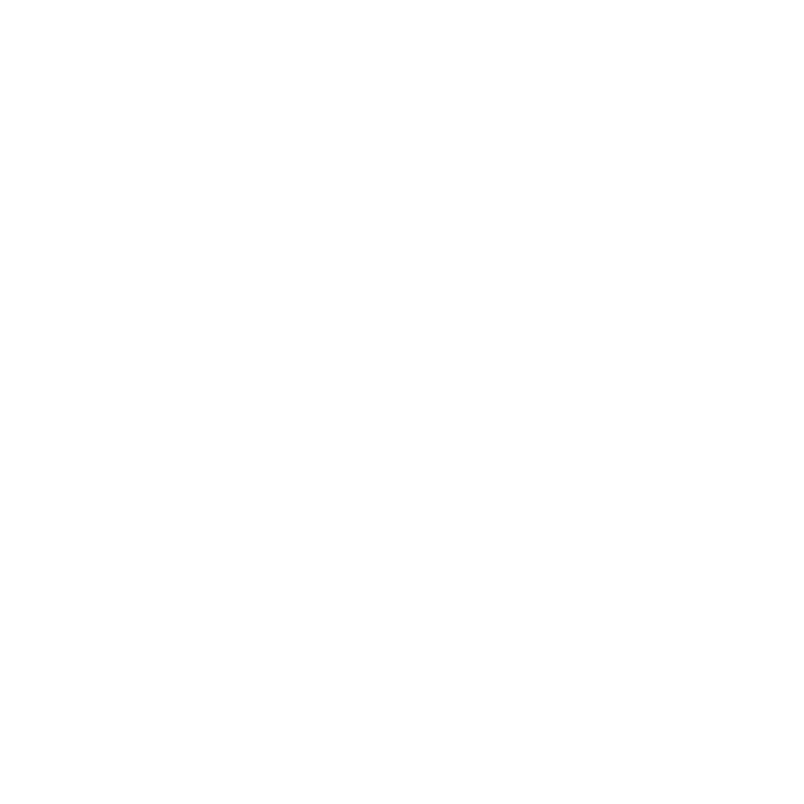
For any state , let denote the probability that starting at state , the process will ever re-enter state .

For **recurrent states**, , meaning every time we leave state , we are guaranteed to re-enter it.

For **transient states**, , meaning there is some path after leaving state that, if we follow it, we will not re-enter state .

Suppose we have the following transition matrix:

The **transition diagram** for this matrix looks like this:



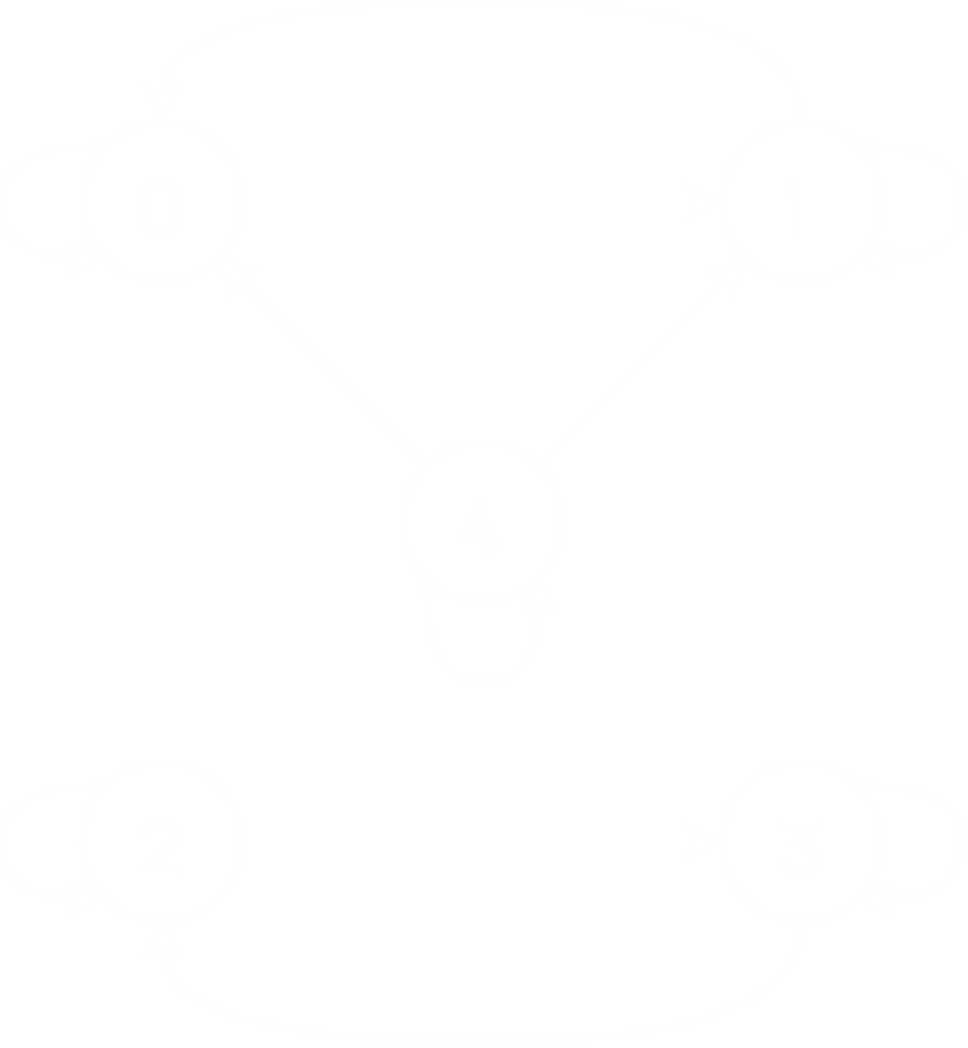
From the transition diagram, we can obtain the list of classes, which is just one item, . An item in the list of classes is a set of states that communicate with each other.

If one state in a class is recurrent, then they all are. For the above diagram, we can clearly see that the state is recurrent. Thus, , and are also recurrent.

## Irreducible Markov Chains

An **irreducible Markov Chain** is one in which every state can be reached from every other state in a finite no. of steps.

Exercises



Classes: , ,

and are recurrent because is recurrent.

and are recurrent because is recurrent.

is transient.

If it rains today, the probability that it will rain tomorrow is . If it does not rain today, the probability that it will rain tomorrow is . What is the probability that it will rain in 4 days?

Answer:

Suppose we have a scenario where the probability of it raining tomorrow depends on both whether it rained today and whether it rained tomorrow. In this case, we should consider the Markov chain to behave like sliding window. Whether it rained yesterday and today affects whether it rains today and tomorrow. This is a single slide for the window (thus we find ). This in turn affects whether it rains tomorrow and the day after, which is another slide (thus we find ).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | Today | Tomorrow | |  | |  | |  | |  | |  | |  | |
|  | Yesterday | Today | R | R | R | | N | | N | | R | | N | | N | |
|  | R | R | 0.7 | | 0.3 | | | | 0 | | | | 0 | | | |
| R | N | 0 | | 0 | | | | 0.4 | | | | 0.6 | | | |
| N | R | 0.5 | | 0.5 | | | | 0 | | | | 0 | | | |
| N | N | 0 | | 0 | | | | 0.2 | | | | 0.8 | | | |

If yesterday was Monday and today is Tuesday, we can thus find the probability of it raining on Thursday. However, note that we do not yet know whether it rains on Wednesday, so we need to take both the probability of it raining and not raining on Wednesday into account.

Thus, the answer is .

## Long-Run Probabilities

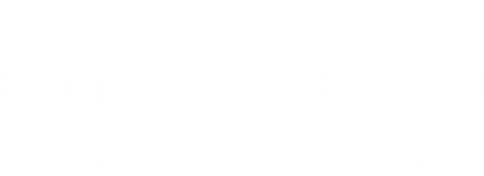
As we keep increasing the power of , we will start to find that up to a small number of decimal places, the values do not change. For example, we could have situations where . At this point, it not longer makes sense to keep doing increasingly lengthy matrix multiplications. Instead, we can use something called the **long-run probability**.

Suppose we have the following transition matrix:

Since this is a 2x2 matrix, we know that there are two states. As such, there will be 2 long-run probabilities, one for each state. These are given by

## Limiting Probability

**Limiting Probabilities** are frequently exactly the same as long-run probabilities. The only cases where they are different is when we have cycles.



For the cycle above, if our stating state is state 0, it is impossible that we end up in state 0 after an odd number of turns, so , and so on. On the other hand, we are guaranteed to be in state 1 after an even number of turns, so , and so on. Because of this pattern, we cannot use the long run probability. Instead, we use this formula:

A similar pattern can be established for cycles over more states. For example, if we have a cycle with three states, only if .

Questions

a)

Answer: 0.73

b)

Limiting Probabilities:

Long Run Probabilities:

State 0

State 1

State 2

The long run probabilities are equal to the limiting probabilities when there are no cycles.

## Gambler’s Ruin Problem

Suppose Max and Patty decide to flip pennies with the one coming closest to the wall winning. Patty, being the better player, has a probability 0.6 of winning on each flip.

If Patty starts with five pennies and Max with ten, what is the probability that Patty will wipe Max out?

The formula above is only used when . Otherwise, we simply use .

### Proof

We know that , and .

For , []

For ,

For ,

Taking the sum of all of the above,

Removing ,

For ,

Example

Suppose Alex has 4 pennies and Sam has 10 pennies. What is the probability that Alex will wipe out Sam?

Example

What is the probability of each gene pair, AA, Aa and aa, appearing in successive generations?

In generation 0, let the probability of the gene pair AA appear be , the probability of the gene pair Aa appearing be and the probability of the gene pair aa appearing be .

For generation 1,

For generation 2,

Thus, the gene pool remains the same in successive generations.

This is called the **Hardy-Weinberg Law**, which states that all successive generations of a population have the gene pairs AA, Aa and aa remains fixed.

This assumes two things:

1. Mating is random.
2. There are no mutations.

Example

This example deals with the **Bonus Malus System**. Suppose an insurance company has four plans, with the premiums $200, $250, $400 and $600 respectively. If a custom is on the th plan and makes claims in a given year, they will automatically be moved to the th plan if , and will be moved to the th plan if . The transition matrix looks like this:

where is the probability of making claims.

Suppose that for a particular customer, the probability values are a Poisson distribution with . Thus,

Our goal is to find the average annual payout that must me made to this customer

From the transition matrix, we can calculate the long-run probabilities.

From this, the average annual payout is given by

## Acceptable and Unacceptable States

There are conditions under which a particular state could be considered to be **unacceptable**. In the previous example, if some state has a very high premium, customers will not want to be in that state and might stop buying the service from the company. Such a state is thus unacceptable. The criteria for unacceptability depends on the company.

It is frequently useful to calculate the proportion of time for which we are in an unacceptable or acceptable state. Suppose we have the following transition matrix:

We are told that states 1 and 2 are acceptable and states 3 and 4 are unacceptable.

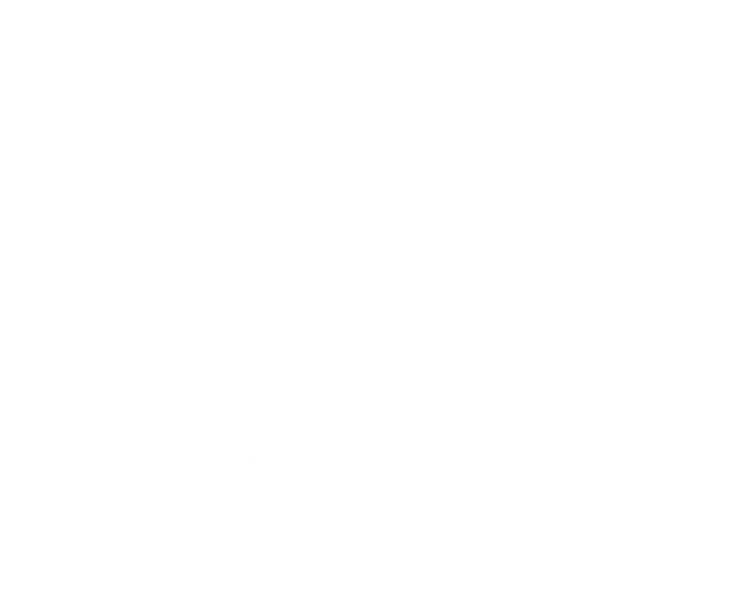
We can calculate the **rate of change** from acceptable to unacceptable states as:

That is, for each acceptable state, we multiply the long-run probability of that state with the sum of the probabilities of going to each of the unacceptable states from that state. This is also called the **rate of breakdown** or the **rate of failure**.

To find the average time spent in the acceptable states:

## Markov Decision Processes

A **Markov Decision Process** is a natural extension of Markov changes. In an MDP, we can draw a state diagram in which different actions state changes. However, each **action** has some **probability** associated with it, and depending on that probability, we will go to a new state. The change of state will also give a **reward**.



For example, if we are in the low state and execute the search action, we will end up back in the low state with a probability and get a reward as a result.

If we know the transition probabilities, this is a simple enough computation. However, what if the transition probabilities are unknown? In this case, the probabilities must be found by trial and error. This is a process called **reinforcement learning**.