INFO510FA24-AB-1

Aditya Bandimatt

2024-11-27

Markov Chain

Setup and load libraries The library(expm) is loaded for matrix exponentiation functions which help us analyze long-term behaviors of Markov chains by raising the transition matrix to higher powers.

```r
# Define the transition matrix for our weather model
P <- matrix(c(0.6, 0.3, 0.1,    # From Sunny
              0.4, 0.4, 0.2,    # From Cloudy
              0.2, 0.3, 0.5),   # From Rainy
            nrow = 3, byrow = TRUE)

colnames(P) <- c("Sunny", "Cloudy", "Rainy")
rownames(P) <- c("Sunny", "Cloudy", "Rainy")

# Display transition matrix
P
```

```
##        Sunny Cloudy Rainy
## Sunny    0.6    0.3   0.1
## Cloudy   0.4    0.4   0.2
## Rainy    0.2    0.3   0.5
```

We define the transition matrix P where each entry represents the probability of moving from one state (weather condition) to another. The rows correspond to the current state, and the columns correspond to the next state

```r
# Check for absorbing states (a state where once entered, you cannot leave)
is_absorbing <- function(P) {
  diag(P) == 1 & rowSums(P != 0) == 1
}

absorbing_states <- which(is_absorbing(P))
if (length(absorbing_states) == 0) {
  print("There are no absorbing states in the weather model.")
} else {
  print(paste("The absorbing states are:", colnames(P)[absorbing_states]))
}
```

```
## [1] "There are no absorbing states in the weather model."
```

The function is_absorbing checks for absorbing states. An absorbing state is one where once entered, you cannot leave (i.e., the diagonal entry is 1 and all other transitions from this state are 0). In our weather model, no weather state is absorbing since weather changes continuously, so no state will have 1 on the diagonal with no exits.

```r
# Identify Recurrent and Transient States
# Calculate long-term behavior by raising the matrix to a large power
P_100 <- P %^% 100  # Raise to the 100th power to check for long-term stability

# Check if the probabilities converge (recurrent state) or diminish (transient)
is_recurrent <- function(P) {
  apply(P > 0, 1, all)
}

recurrent_states <- which(is_recurrent(P_100))
transient_states <- setdiff(1:nrow(P), recurrent_states)

# Output results
print(paste("The recurrent states are:", colnames(P)[recurrent_states]))
```

```
## [1] "The recurrent states are: Sunny"  "The recurrent states are: Cloudy"
## [3] "The recurrent states are: Rainy"
```

```r
print(paste("The transient states are:", colnames(P)[transient_states]))
```

```
## [1] "The transient states are: "
```

Here, we raise the transition matrix P to the power of 100 (P_100 <- P %^% 100) to simulate the long-term behavior of the Markov chain This gives us insight into whether the chain stabilizes and reveals recurrent states (those that are revisited infinitely often). We define a state as recurrent if, after a large number of steps, the probability of returning to that state is still greater than zero (is_recurrent). The transient states are those that are not revisited with certainty over the long run

```r
# Check if a state is periodic
gcd_period <- function(x) {
  if (length(x) > 1) {
    return(Reduce(gcd.bigz, x))# Use Reduce to compute gcd across multiple values
  } else {
    return(x)
  }
}

# Check if a state is periodic
is_periodic <- function(P, state, steps = 100) {
  powers <- lapply(1:steps, function(k) P %^% k)
  periods <- sapply(powers, function(M) M[state, state] > 0)
  period_indices <- which(periods)# Get the step numbers where state is revisited
  if (length(period_indices) > 1) {
    gcd_value <- gcd_period(period_indices)  # Compute GCD of step numbers
    return(gcd_value > 1)  # A periodic state has a GCD greater than 1
  } else {
    return(FALSE)
  }
}

# Check for periodic states
periodic_states <- sapply(1:nrow(P), function(s) is_periodic(P, s))
if (any(periodic_states)) {
  print(paste("The periodic states are:", colnames(P)[which(periodic_states)]))
} else {
  print("There are no periodic states in the weather model.")
}
```

```
## [1] "There are no periodic states in the weather model."
```

We check if a state is periodic by observing the number of steps in which a return to that state occurs and calculating the greatest common divisor (GCD) of those step counts. If the GCD is greater than 1, the state is periodic (i.e., revisited in regular intervals). In our weather model, it's unlikely that weather follows a strict period, so we expect no periodic states.

```r
# Simulate the weather evolution over 10 days
weather_states <- c("Sunny", "Cloudy", "Rainy")
set.seed(123)  # Set a seed for reproducibility

simulate_weather <- function(P, days = 10, start_state = 1) {
  weather_seq <- numeric(days)
  weather_seq[1] <- start_state

  for (i in 2:days) {
    weather_seq[i] <- sample(1:3, size = 1, prob = P[weather_seq[i-1], ])
  }

  return(weather_states[weather_seq])
}

simulated_weather <- simulate_weather(P, days = 100, start_state = 1)
print(paste("Simulated weather sequence over 10 days:",
            paste(simulated_weather, collapse = " -> ")))
```

```
## [1] "Simulated weather sequence over 10 days: Sunny -> Sunny -> Cloudy -> Sunny -> Cloudy -> Rainy -
```

To better understand the dynamics of the Markov chain, we simulate the weather for 100 days starting with a Sunny day. # The function simulate_weather selects a next state based on the transition matrix probabilities. The output shows a potential sequence of weather conditions over time, illustrating the probabilistic nature of the transitions.

Conclusion

By analyzing the weather model using a Markov chain, we classified the states into different categories:

Absorbing states: None in this case. Recurrent states: These are the states we expect to revisit often. Transient states: These are states that may disappear after many steps. Periodic states: States revisited in regular cycles.

This type of classification helps in understanding long-term patterns and stability in systems modeled by Markov chains.