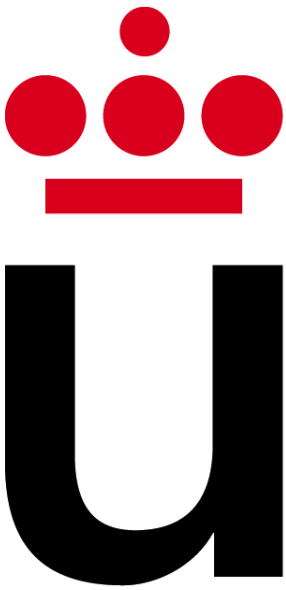
Caso práctico III

Simulación y Metaheurísticas

Análisis de resultados

José Ignacio Escribano



Móstoles, 22 de mayo de 2016

Índice de tablas

Índice de figuras

Índice

1. Introducción

Este caso práctico veremos cómo simular una cadena de Markov en tiempo discreto.

2. Resolución del caso práctico

A continuación resolveremos cada una de las cuestiones planteadas.

2.1. Cuestión 1

Sabemos que la matriz de transición de la cadena de Markov tiene los siguientes valores:

$$P = \begin{pmatrix} 0.1 & 0.4 & p_{13} \\ 0.1 & 0.7 & p_{23} \\ 0.3 & 0.1 & p_{33} \end{pmatrix}$$

donde p_{13}, p_{23} y p_{33} son valores desconocidos.

Puesto que la matriz P debe ser una matriz estocástica, esto es, la suma de sus filas debe ser 1, aplicamos esta restricción a cada una de las filas, quedando un sistema como el que sigue:

$$\begin{aligned} 0.1 + 0.4 + p_{13} = 1 &\iff p_{13} = 0.5 \\ 0.1 + 0.7 + p_{23} = 1 &\iff p_{23} = 0.2 \\ 0.3 + 0.1 + p_{33} = 1 &\iff p_{33} = 0.6 \end{aligned}$$

Así pues, la matriz de transición de la cadena de Markov con espacio de estados $S = \{$estodo0, estado1, estado2$\}$ queda de la siguiente forma:

$$P = \begin{array}{c} \\ \text{estado0} \\ \text{estado1} \\ \text{estado2} \end{array} \begin{array}{c} \begin{array}{ccc} \text{estado0} & \text{estado1} & \text{estado2} \end{array} \\ \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.1 & 0.7 & 0.2 \\ 0.3 & 0.1 & 0.6 \end{pmatrix} \end{array}$$

La Figura 1 muestra la matriz de transición en forma de grafo.

2.2. Cuestión 2

Usamos el algoritmo visto en clase para generar transiciones de la cadena de Markov. El código implementado se puede ver en el Anexo A.

Haciendo 100 réplicas con 1000 muestras de la cadena de Markov cada una, se obtienen los resultados de la Tabla 1.

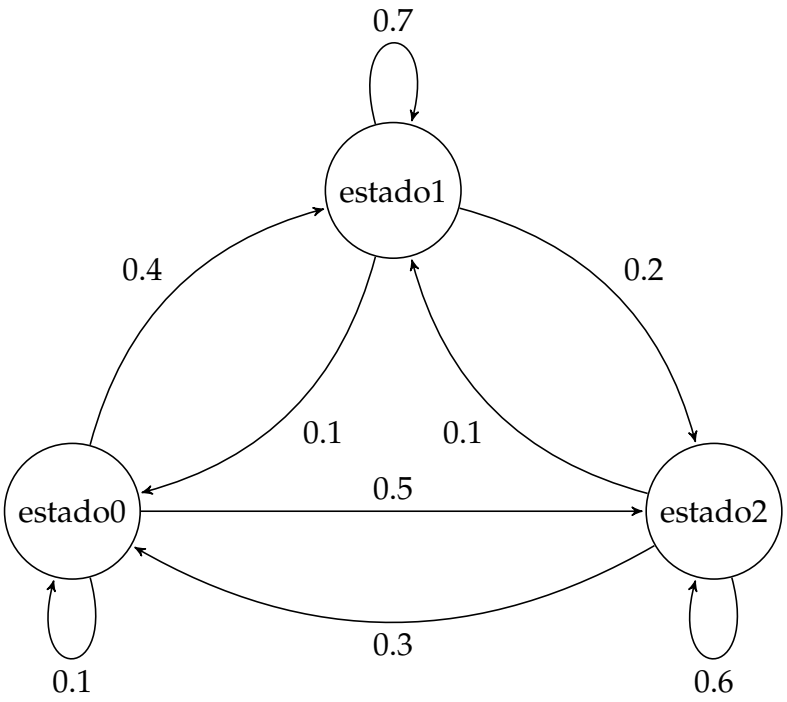


Figura 1: Cadena de Markov en forma de grafo

Tabla 1: Media y desviacón típica de cada estado

Estado	**Media**	**Desviación típica**
estado0	0.2933	0.035
estado1	0.3068	0.042
estado2	0.3998	0.041

Es decir, el computador estará en el 29.33 % del tiempo en el estado0, el 30.68 % en el estado1 y el 39.98 % en el estado2.

Podemos conocer la solución analítica, aplicando potencias a la matriz de transición de la cadena de Markov, es decir, P^2, P^3, ..., P^{100}, ... ya que cuando $\lim_{k\to\infty} P^k$ cada fila de P tiene al vector estacionario de la cadena de Markov. Así pues, estos son algunos valores de las potencias:

$$P^2 = \begin{pmatrix} 0.20 & 0.37 & 0.43 \\ 0.14 & 0.55 & 0.31 \\ 0.22 & 0.25 & 0.53 \end{pmatrix}$$

$$P^7 = \begin{pmatrix} 0.1853320 & 0.3883624 & 0.4263056 \\ 0.1835368 & 0.3947920 & 0.4216712 \\ 0.1866264 & 0.3837280 & 0.4296456 \end{pmatrix}$$

$$P^{22} = \begin{pmatrix} 0.1851851924 & 0.3888888629 & 0.4259259446 \\ 0.1851851040 & 0.3888891798 & 0.4259257163 \\ 0.1851852562 & 0.3888886346 & 0.4259261092 \end{pmatrix}$$

$$P^{52} = P^{57} = \cdots = P^{102} = \begin{pmatrix} 0.1851851852 & 0.3888888889 & 0.4259259259 \\ 0.1851851852 & 0.3888888889 & 0.4259259259 \\ 0.1851851852 & 0.3888888889 & 0.4259259259 \end{pmatrix}$$

Así pues el vector estacionario es $(0.185, 0.3888, 0.4259)$. El método implementado sobrestima el tiempo en el estado0 y lo subestima en el estado1. El estado2 es bastante preciso.

Calculamos intervalos de confianza para al 95 % para cada uno de los estados. Así pues,

- Estado 0: $[0.2714, 0.3151]$
- Estado 1: $[0.2805, 0.3330]$
- Estado 2: $[0.3751, 0.4244]$

Con nivel de confianza del 95 %, podemos asegurar que los valores de cada uno de estos tres estados estarán dentro de cada intervalo de confianza.

3. Conclusiones

En este caso práctico hemos visto cómo simular cadenas de Markov en tiempo discreto, una herramienta muy útil utilizada en diversas áreas.

A. Código R utilizado

A continuación se muestra el código utilizado para la realización de este caso práctico.

```
#################################################################
# Cuestión 1
#################################################################

# Matriz de transici?n de la cadena de Markov
P = matrix(c(0.1, 0.4, 0.5, 0.1, 0.7, 0.2, 0.3, 0.1, 0.6),
           nrow = 3,
           ncol = 3,
           byrow = TRUE)


# Nombre de los estados
states = c("estado0", "estado1", "estado2")

# Asignamos los estados a la matriz de transición
dimnames(P) <- list(states, states)


# Definimos una función para muestrear una cadena de Markov
samplingCMTD = function(P, N, i0){
  X = array()
  X[1] = i0
  i = 2
  while(i < N){
    h = rgeom(1, 1-P[i0,i0])
    #cat("h:", h, "\n i:",i, "\n", "i+h:", i+h, "\n")
    X[i+h] = generateDist(P, i0, colnames(P))
    #cat("estado: ", X[i+h], "\n", "---------------\n")
    i0 = X[i+h]
    i = i + 1
  }
  #X = X[!is.na(X)]
  return(X)
}

# Definimos una función para generar la distribución discreta
generateDist = function(P, state, states){

  # Eliminamos el estado en el que nos encontramos
  states = setdiff(states, state)
```

```
  # Vector de probabilidades vac?o
  probs = array(dim = length(states))

  # Recorremos el vector de estados
  # y asignamos la probabilidad
  for(i in 1:length(states)){
    probs[i] = P[state, states[i]]/(1-P[state, state])
  }

  names(probs) = states

  u = runif(1)

  # Calculamos el vector acumulado
  # y añadimos 0 al comienzo
  cumulative = unname(c(0,cumsum(probs)))

  # Calculamos la longitud real del vector (le hemos añadido una posición nueva)
  n = length(cumulative) - 1

  # Buscamos la posición que acumula la cantidad de probabilidad
  # generada por el número aleatorio y lo asignamos al estado.
  for(i in 1:n){
    if((cumulative[i] <= u) & (u <= cumulative[i+1])){
      stat = states[i]
      break
    }
  }

  return(stat)

}

#####################################################################
# Cuestión 2
#####################################################################

# Número de muestras de cada réplica
N = 1000

# Número de réplicas
n = 100
```

```r
X = list()
for(i in 1:n){
  a = samplingCMTD(P, N, "estado0")
  X[[i]] = summary(as.factor(na.omit(a)))/length(na.omit(a))
}

# Guardamos la media de las proporciones de cada estado
state1 = array()
state2 = array()
state3 = array()

for(i in 1:n){
  state1[i] = X[[i]][[1]]
  state2[i] = X[[i]][[2]]
  state3[i] = X[[i]][[3]]
}

# Media y desviación típica de cada estado
mean(state1)
sd(state1)

mean(state2)
sd(state2)

mean(state3)
sd(state3)

# Cargamos el paquete expm y si no esta lo instalamos
if(!require("expm")){install.packages("expm")}
require("expm")

for(i in seq(2,102, 5)){
  cat("P^", i, "\n")
  print(P%^%i)
}
```