**Científicos de Datos**

**TP 1 - Ejercicio práctico IO**

Integrantes:

Bertino, Ariel

Manzalini, Andrés

LU:

247865

247729

Mails:

[abertino@alumnos.exa.unicen.edu.ar](mailto:abertino@alumnos.exa.unicen.edu.ar)

[amanzalini@alumnos.exa.unicen.edu.ar](mailto:abertino@alumnos.exa.unicen.edu.ar)

Grupo: 5

Docentes:

Illescas, Gustavo

Mora Soto, José Arturo

**Ejercicio elegido de Investigación operativa:**

Trabajo Práctico No 4: Teoría de Decisiones (Parte III) - Juegos

7. Representar con una matriz de juegos el clásico juego “Piedra-Papel-Tijera”. ¿Es un

juego de suma cero? ¿Tiene solución por estrategias puras? Simular una iteración del juego.

**Desarrollo:**

Para generar las instancias de prueba creamos dos vectores, uno por cada jugador, de 100 jugadas. Les aplicamos una función aleatoria que si sale entre (0; 0.33] es piedra, si sale entre (0.33; 0.66] es papel y por último si se obtiene un valor entre (0.66; 1) se le asignará piedra.

Una vez generados los valores aleatorios que representan las jugadas, tenemos dos vectores de jugadas para hacer la simulación.

Generamos la matriz de juegos como una matriz de 3x3 de piedra, papel y tijera.

Y procesamos los vectores para ir cargando la matriz de juego a medida que se hace el procesamiento.

Es un juego simultáneo de suma 0 porque lo que gana un jugador y es inversamente proporcional a lo que gana el otro.En caso de empatar el estado sigue igual.

Las posibles ganancias: -1, 0 y 1

Perspectiva del J1 en la siguiente tabla de ganancias

|  |  |  |  |
| --- | --- | --- | --- |
| J2/J1 | Piedra | Papel | Tijera |
| Piedra | 0 | 1 | -1 |
| Papel | -1 | 0 | 1 |
| Tijera | 1 | -1 | 0 |

Tiene solución por estrategia pura porque las estrategias que se pueden elegir son piedra, papel o tijera.

Aunque también puede considerarse como de estrategias mixtas en caso de utilizar el patrón de juego del adversario como estrategia.

A continuación presentamos el código en lenguaje R que representa el juegos en consideración

#VALOR CONSTANTES A LO LARGO DEL CODIGO

cantJugadas <- 100

Piedra = "PIEDRA"

Papel = "PAPEL"

Tijera = "TIJERA"

#install.packages("ggplot2")

library(ggplot2)

#matriz de probabilidades resultante

matResultados <- matrix(0, nrow = 3, ncol = 3, byrow = FALSE)

#PIEDRA=1, PAPEL=2, TIJERA=3

#definimos vectores para las jugadas PIEDRA-PAPEL-TIJERA

jugadas1 <- vector(mode = "character", length = cantJugadas)

jugadas2 <- vector(mode = "character", length = cantJugadas)

#inicializamos dos vectores j1y j2 con valores aleatorios entre 0 y 1

j1 <- runif(cantJugadas, min=0, max=1)

j2 <- runif(cantJugadas, min=0, max=1)

#funcion que carga el vJR(vector de jugadas reales) con piedra papel o tijera

#basado en el vJA(vector de jugadas aleatorias)

cargarVectorJugadas <- function(vJA, vJR){

for(i in 1:cantJugadas){

if ((vJA[i] >= 0) & (vJA[i] <= .33)){

vJR[i] <- Piedra

}

else

if ((vJA[i] > 0.33) & (vJA[i] <= 0.66)){

vJR[i] <- Papel

}

else

if ((vJA[i] > 0.66) & (vJA[i] <= 1)){

vJR[i] <- Tijera

}

}

return(vJR)

}

#funcion que simular el juego en si

jugar <- function (jug1,jug2,matResul){

for(i in 1:cantJugadas){

if (jug1[i] == jug2[i]){

if (jug1[i] == Piedra){

matResul[1,1] <- matResul[1,1] + 1

} else if (jug1[i] == Papel){

matResul[2,2] <- matResul[2,2] + 1

} else if (jug1[i] == Tijera){

matResul[3,3] <- matResul[3,3] + 1

}

print(paste("Jugador1 y Jugador2 sacaron lo mismo:",jug1[1],", entonces es empate"))

} else if ((jug1[i] == Piedra) & (jug2[i] == Tijera) ){

matResul[1,3] <- matResul[1,3]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador1"))

} else if ((jug1[i] == Papel) & (jug2[i] == Piedra) ){

matResul[2,1] <- matResul[2,1]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador1"))

} else if ((jug1[i] == Tijera) & (jug2[i] == Papel) ){

matResul[3,2] <- matResul[3,2]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador1"))

} else if ((jug2[i] == Piedra) & (jug1[i] == Tijera) ){

matResul[3,1] <- matResul[3,1]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador2"))

} else if ((jug2[i] == Papel) & (jug1[i] == Piedra) ){

matResul[1,2] <- matResul[1,2]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador2"))

} else if ((jug2[i] == Tijera) & (jug1[i] == Papel) ){

matResul[2,3] <- matResul[2,3]+1

print(paste("Jugador1:" ,jug1[i],", Jugador2:",jug2[i], ", ganador: jugador2"))

}

print("---------------------------------------------------------------------------")

}

matResul <- matResul/cantJugadas #genera matris de probabilidades

return(matResul)

}

# print(j1) se carga bien

# print(j2) se carga bien

jugadas1 <- cargarVectorJugadas(j1,jugadas1)

jugadas2 <- cargarVectorJugadas(j2,jugadas2)

# print(jugadas1) se carga bien

# print(jugadas2) se carga bien

matResultados <- jugar(jugadas1, jugadas2, matResultados)

print(matResultados)

# matResultadosDf <- as.data.frame(t(matResultados)) #transforma una matriz en un dataFrame de R

Por lo tanto, generamos dos vectores con jugadas aleatorias, jugadas1 y jugadas2

Cada vector tiene 100 jugadas de 3 valores posibles: piedra, papel o tijera

Con esta información de las jugadas podemos completar la matriz de juego de 3x3 con las probabilidades resultantes de la simulación.

**Librerías aplicadas:**

En un principio estos vectores simples usan la función de R base, runif(cantJugadas), que genera números aleatorios entre 0 y 1.

Para procesarlos adecuadamente, estos vectores podemos pasarlos a formato data.frame -> aplicamos la librería ***base*** de R, la de data.frame, pero que en este caso no representa

ba un cambio sustancial y/o apreciable en la visualización de los datos.

*dataset <- data.frame("J1"=jugadas1, "J2"=jugadas2)*

De esta manera transformamos los datos a una estructura del tipo data.frame, al cual podemos aplicarle distintas funciones.

Como por ejemplo graficarlo, con la librería ***ggplot()***

Un enfoque relacionado con machine learning es aplicarle al dataset una regresión lineal o una clasificación con lalibrería ***CARET,*** o cualquier otro modelo disponible en la librería.

Cabe aclarar que si bien es posible aplicar técnicas de machine Learning al problema en este trabajo desarrollado, son técnicas que se aplican sobre el modo de jugar o la frecuencia en que determinado jugador elige un elemento dado durante la partida.

**Bibliografía y conclusión**

Enfoque sobre juegos en machine learning:

<https://www.kdnuggets.com/2020/03/crash-course-game-theory-machine-learning.html>

Enfoque sobre el piedra papel o tijera, sobre juegos simultáneos:

<https://www.kdnuggets.com/2018/06/introduction-game-theory-part-1.html>

A simple vista nos pareció lo más simple de hacer, pero después fueron apareciendo distintas alternativas, distintos enfoques de machine learning que no eran tan simples como nos imaginamos en un principio.

Como por ejemplo redes neuronales para procesamiento de imagen, procesando por la cámara las imágenes de la mano e identificando si es piedra, papel o tijera.

O redes neuronales adversarias que intentan engañarse mutuamente, buscando el patrón de aleatoriedad del jugador.

En fin, hay muchos posibles enfoques para un mismo problema, y por mas simple que parezca el problema, siempre existe la forma de agregarle complejidad.