battleship_AI

Juliana Quirós, Alberto (ajuliana@ucm.es)

Script para el sistema de ataque del juego "hundir la flota". Desarrollo:

1) Pido a Chatgpt que me facilite un listado de buenas (basadas en estrategia) y malas jugadas intercaladas dada una flota concreta. Creo un vector de aprendizaje supervisado (0 malas, 1 buenas)
2) Preproceso los datos a formato matricial, descomponiendo la flota en posiciones en eje x y eje y, y uniendo a estas 2 columnas la de jugadas (eje x y eje y) y el vector de aprendizaje supervisado.
3) Entreno una red con dicha matriz
4) Creo nuevo set de entrenamiento, y entreno otra red, tomando los pesos de la anterior como starting weights.
5) Repito hasta 4 veces con la misma flota y diferentes jugadas.
6) Cambio a otro tipo de flota y repito el proceso 2 veces.
7) La matriz de confusión es catastrófica (probablemente debido al poco entrenamiento. Empleé 108 jugadas entre entrenamiento y test)
8) Mi red en este punto, debería dar un listado de buenas y malas jugadas.
9) La estrategia a partir de ahora es la siguiente: se le pregunta al jugador si ha alcanzado al enemigo tras un disparo aleatorio; si la respuesta es "no", la función lanzamiento utilizará la lista de mejores jugadas en la que la red ha sido entrenada para hacer el siguiente disparo, empleando todo el tablero como área de ataque. Por otra parte, si la respuesta es "sí", acotará el área de ataque a un rango alrededor del punto alcanzado, empleando su lista de mejores jugadas en este rango.

   La estrategia confía en la extrapolación del aprendizaje, suponiendo que si la red es capaz de dar las mejores jugadas en un área de ataque que comprende todo el tablero, también las dará en las nuevas áreas acotadas tras el impacto.
   Esta estrategia, sin embargo, tiene un coste muy alto: las buenas jugadas se repiten, y si además, acotas el rango, se repiten aún más. Para solventarlo, he introducido algunos mecanismos aleatorios de selección del rango, aumentando el número de valores a pasar a la IA en el eje x y en el y, y seleccionando valores subóptimos del listado de mejores jugadas, pero aun así terminan repitiéndose muchas. ¿Y por qué no aleatorizo el disparo cada vez que se repite un número? Porque entonces de nada serviría haber entrenado la IA en mejores jugadas.
   Al final del script incluyo un ejemplo de partida.

```
library(beepr)
library(neuralnet)
source("E:\\GDrive1\\Uni\\Master\\simulacion\\ejercicios\\battleship\\game\\funciones.R")

############## Training set 1#######################################
flota <- list()

flota[[1]] <- creaBarco("A", c(1, 1, 2, 1, 3, 1))

flota[[2]] <- creaBarco("B", c(5, 1, 5, 2, 5, 3))

flota[[3]] <- creaBarco("C", c(6, 3, 7, 3, 8, 3))
```

```r
tablero <- nuevoTablero(10, 10, flota)

## Le pido a chatgpt que genere jugadas para la flota anterior empleando
## búsqueda en cruz y que intercale malas jugadas
jugadas <- list()
## Subset 1
jugadas[[1]] <- c(1, 1)  # buena jugada
jugadas[[2]] <- c(2, 1)  # buena jugada
jugadas[[3]] <- c(3, 1)  # buena jugada
jugadas[[4]] <- c(4, 1)  # buena jugada
jugadas[[5]] <- c(5, 1)  # buena jugada
jugadas[[6]] <- c(6, 1)  # mala jugada
jugadas[[7]] <- c(7, 1)  # buena jugada
jugadas[[8]] <- c(8, 1)  # buena jugada
jugadas[[9]] <- c(9, 1)  # buena jugada
## Subset 2
jugadas[[10]] <- c(10, 1)  # mala jugada
jugadas[[11]] <- c(10, 2)  # buena jugada
jugadas[[12]] <- c(10, 3)  # buena jugada
jugadas[[13]] <- c(10, 4)  # mala jugada
jugadas[[14]] <- c(9, 4)  # buena jugada
jugadas[[15]] <- c(8, 4)  # mala jugada
jugadas[[16]] <- c(8, 5)  # buena jugada
jugadas[[17]] <- c(8, 6)  # buena jugada
jugadas[[18]] <- c(8, 7)  # buena jugada
## Subset 3
jugadas[[19]] <- c(8, 8)  # buena jugada
jugadas[[20]] <- c(8, 9)  # mala jugada
jugadas[[21]] <- c(7, 9)  # buena jugada
jugadas[[22]] <- c(6, 9)  # buena jugada
jugadas[[23]] <- c(5, 9)  # mala jugada
jugadas[[24]] <- c(5, 8)  # buena jugada
jugadas[[25]] <- c(5, 7)  # buena jugada
jugadas[[26]] <- c(5, 6)  # mala jugada
jugadas[[27]] <- c(4, 6)  # buena jugada
## Test subset
jugadas[[28]] <- c(3, 6)  # buena jugada
jugadas[[29]] <- c(2, 6)  # buena jugada
jugadas[[30]] <- c(1, 6)  # mala jugada
jugadas[[31]] <- c(1, 7)  # buena jugada
jugadas[[32]] <- c(1, 8)  # buena jugada
jugadas[[33]] <- c(1, 9)  # mala jugada
jugadas[[34]] <- c(2, 9)  # buena jugada
jugadas[[35]] <- c(3, 9)  # mala jugada
jugadas[[36]] <- c(3, 8)  # buena jugada
## No utilizados
jugadas[[37]] <- c(3, 7)  # buena jugada
jugadas[[38]] <- c(3, 6)  # mala jugada
jugadas[[39]] <- c(4, 5)  # mala jugada
jugadas[[40]] <- c(5, 5)  # mala jugada


## Defino vector indicando cuáles son buenas
```

```r
buena_jugada <- c(1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
    1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0)




############## Preprocess####################### Extrae coordenadas de posiciones


posiciones_y <- c()
posiciones_x <- c()
c = 1
for (a in 1:length(flota)) {
    for (n in 1:nrow(flota[[a]]$pos)) {
        posiciones_y[c] <- flota[[a]]$pos[n, 1]
        posiciones_x[c] <- flota[[a]]$pos[n, 2]
        c = c + 1
    }
}

## Extrae coordenadas de jugadas

jugadas <- as.data.frame(jugadas)
jugadas_y <- as.integer(jugadas[1, ])
jugadas_x <- as.integer(jugadas[2, ])




training_data_11 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[1:9],
    jugadas_y = jugadas_y[1:9], buena_jugada = buena_jugada[1:9])
training_data_12 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[10:18],
    jugadas_y = jugadas_y[10:18], buena_jugada = buena_jugada[10:18])
training_data_13 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[19:27],
    jugadas_y = jugadas_y[19:27], buena_jugada = buena_jugada[19:27])
training_data_14 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[28:36],
    jugadas_y = jugadas_y[28:36], buena_jugada = buena_jugada[28:36])
##### Training con esas jugadas y posiciones#################################

nn_11 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
    hidden = 20, data = training_data_11, algorithm = "sag", learningrate = 0.001,
    act.fct = "logistic", rep = 30)


nn_12 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
    training_data_12, hidden = 20, startweights = nn_11$weights, algorithm = "sag",
    learningrate = 0.001, act.fct = "logistic", rep = 30)

nn_13 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
    training_data_13, hidden = 20, startweights = nn_12$weights, algorithm = "sag",
    learningrate = 0.001, act.fct = "logistic", rep = 30)

nn_14 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
```

```
    training_data_14, hidden = 20, startweights = nn_13$weights, algorithm = "sag",
    learningrate = 0.001, act.fct = "logistic", rep = 30)
```

```
############## Training set 2######################

flota <- list()

flota[[1]] <- creaBarco("A", c(1, 1, 1, 2, 1, 3, 1, 4))

flota[[2]] <- creaBarco("B", c(3, 1, 4, 1, 5, 1, 6, 1))

flota[[3]] <- creaBarco("C", c(8, 1, 8, 2, 8, 3))

flota[[4]] <- creaBarco("D", c(3, 3, 3, 4, 3, 5))

flota[[5]] <- creaBarco("E", c(5, 3, 5, 4))

flota[[6]] <- creaBarco("F", c(7, 5, 7, 6))

flota[[7]] <- creaBarco("G", c(9, 3, 9, 4))

flota[[8]] <- creaBarco("H", c(2, 7))

flota[[9]] <- creaBarco("I", c(4, 7))

flota[[10]] <- creaBarco("J", c(6, 7))

flota[[11]] <- creaBarco("K", c(8, 7))

tablero <- nuevoTablero(10, 10, flota)

jugadas <- list()
jugadas[[1]] <- c(1, 1)  # buena jugada
jugadas[[2]] <- c(3, 1)  # mala jugada
jugadas[[3]] <- c(2, 1)  # buena jugada
jugadas[[4]] <- c(4, 1)  # mala jugada
jugadas[[5]] <- c(3, 2)  # buena jugada
jugadas[[6]] <- c(5, 1)  # mala jugada
jugadas[[7]] <- c(4, 2)  # buena jugada
jugadas[[8]] <- c(6, 1)  # mala jugada
jugadas[[9]] <- c(5, 2)  # buena jugada
jugadas[[10]] <- c(7, 1)  # mala jugada
jugadas[[11]] <- c(6, 2)  # buena jugada
jugadas[[12]] <- c(8, 1)  # mala jugada
jugadas[[13]] <- c(7, 2)  # buena jugada
```

```
jugadas[[14]] <- c(9, 1)  # mala jugada
jugadas[[15]] <- c(8, 2)  # buena jugada
jugadas[[16]] <- c(10, 1)  # mala jugada
jugadas[[17]] <- c(9, 2)  # buena jugada
jugadas[[18]] <- c(1, 3)  # mala jugada
jugadas[[19]] <- c(10, 2)  # buena jugada
jugadas[[20]] <- c(2, 3)  # mala jugada
jugadas[[21]] <- c(4, 4)  # buena jugada
jugadas[[22]] <- c(6, 6)  # buena jugada
jugadas[[23]] <- c(8, 8)  # buena jugada
jugadas[[24]] <- c(1, 10)  # buena jugada
jugadas[[25]] <- c(9, 2)  # buena jugada
jugadas[[26]] <- c(3, 7)  # buena jugada
jugadas[[27]] <- c(5, 5)  # buena jugada
jugadas[[28]] <- c(2, 5)  # buena jugada
jugadas[[29]] <- c(7, 8)  # buena jugada
jugadas[[30]] <- c(10, 1)  # buena jugada
jugadas[[31]] <- c(1, 7)  # mala jugada
jugadas[[32]] <- c(9, 9)  # mala jugada
jugadas[[33]] <- c(4, 6)  # mala jugada
jugadas[[34]] <- c(8, 1)  # mala jugada
jugadas[[35]] <- c(2, 10)  # mala jugada
jugadas[[36]] <- c(6, 4)  # mala jugada
jugadas[[37]] <- c(10, 7)  # mala jugada
jugadas[[38]] <- c(3, 2)  # mala jugada
jugadas[[39]] <- c(5, 9)  # mala jugada
jugadas[[40]] <- c(7, 3)  # mala jugada
jugadas[[41]] <- c(8, 8)  # buena jugada
jugadas[[42]] <- c(2, 9)  # mala jugada
jugadas[[43]] <- c(7, 1)  # buena jugada
jugadas[[44]] <- c(9, 9)  # mala jugada
jugadas[[45]] <- c(3, 3)  # buena jugada
jugadas[[46]] <- c(6, 6)  # buena jugada
jugadas[[47]] <- c(4, 4)  # buena jugada
jugadas[[48]] <- c(9, 4)  # mala jugada
## Test####
jugadas[[49]] <- c(7, 9)  # mala jugada
jugadas[[50]] <- c(5, 4)  # mala jugada
jugadas[[51]] <- c(1, 5)  # buena jugada
jugadas[[52]] <- c(3, 9)  # mala jugada
jugadas[[53]] <- c(6, 3)  # buena jugada
jugadas[[54]] <- c(2, 2)  # buena jugada
jugadas[[55]] <- c(4, 6)  # mala jugada
jugadas[[56]] <- c(10, 5)  # mala jugada
jugadas[[57]] <- c(8, 7)  # buena jugada
jugadas[[58]] <- c(9, 6)  # mala jugada
jugadas[[59]] <- c(1, 8)  # mala jugada
jugadas[[60]] <- c(10, 2)  # buena jugada
jugadas[[61]] <- c(3, 5)  # mala jugada
jugadas[[62]] <- c(5, 6)  # buena jugada
jugadas[[63]] <- c(4, 9)  # mala jugada
jugadas[[64]] <- c(6, 8)  # buena jugada
jugadas[[65]] <- c(2, 7)  # mala jugada
```

```r
jugadas[[66]] <- c(7, 5)  # buena jugada
jugadas[[67]] <- c(10, 9)  # mala jugada
jugadas[[68]] <- c(8, 3)  # buena jugada
jugadas[[69]] <- c(9, 4)  # mala jugada
jugadas[[70]] <- c(1, 2)  # buena jugada
jugadas[[71]] <- c(2, 8)  # mala jugada
jugadas[[72]] <- c(3, 10)  # buena jugada

buena_jugada <- c(1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
    0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1)









############## Preprocess####################### Extrae coordenadas de posiciones


posiciones_y <- c()
posiciones_x <- c()
c = 1
for (a in 1:length(flota)) {
    for (n in 1:nrow(flota[[a]]$pos)) {
        posiciones_y[c] <- flota[[a]]$pos[n, 1]
        posiciones_x[c] <- flota[[a]]$pos[n, 2]
        c = c + 1
    }
}

## Extrae coordenadas de jugadas

jugadas <- as.data.frame(jugadas)
jugadas_y <- as.integer(jugadas[1, ])
jugadas_x <- as.integer(jugadas[2, ])




training_data_21 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[1:24],
    jugadas_y = jugadas_y[1:24], buena_jugada = buena_jugada[1:24])
training_data_22 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[25:48],
    jugadas_y = jugadas_y[25:48], buena_jugada = buena_jugada[25:48])
test_data_2 <- data.frame(posiciones_x, posiciones_y, jugadas_x = jugadas_x[49:72],
    jugadas_y = jugadas_y[49:72])
##### Training###############################

nn_21 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
    hidden = 20, startweights = nn_13$weights, data = training_data_21, algorithm = "sag",
    learningrate = 0.001, act.fct = "logistic", rep = 30)
```

```
nn_22 <- neuralnet(buena_jugada ~ posiciones_x + posiciones_y + jugadas_x + jugadas_y,
    hidden = 20, startweights = nn_21$weights, data = training_data_22, algorithm = "backprop",
    learningrate = 0.001, act.fct = "logistic", rep = 30)



res2_training <- predict(nn_22, training_data_22, type = "class")
mse_training2 <- (sum(buena_jugada[25:48] - res2_training)^2)/length(res2_training)
mse_training2

##### TESTING##############################################
res2_test <- predict(nn_22, test_data_2, type = "class")
mse_test2 <- (sum(buena_jugada[49:72] - res2_test)^2)/length(res2_test)
mse_test2

cbind(res2_training, buena_jugada[25:48])
cbind(res2_test, buena_jugada[49:72])



################################################## Toma 24 blancos aleatorios y
################################################## pide que los evalúe la
################################################## red#############



lanzamiento_rand <- function() {
    radar_full <- 1:10  ## Primer lanzamiento, todo el tablero es área de operaciones
    blanco <- c(sample(radar_full, 1), sample(radar_full, 1))
    beep(7)
    print(blanco)
    return(blanco)
}




lanzamiento <- function() {
    randomgen <- runif(1, 0, 1)
    if (alcance_pregunta == 0) {
        rango_x = 1:10
        rango_y = 1:10
    } else {
        ## 2 tipos de exploraciones alrededor del punto, posteriormente
        ## asistida por la IA
        if (randomgen < 0.5) {
            rango_x = c(seq(lanz_ultimo[1] - 4, lanz_ultimo[1] - 1), seq(lanz_ultimo[1] +
                1, lanz_ultimo[1] + 4))
            rango_y = lanz_ultimo[2]
        } else {
            rango_x = lanz_ultimo[1]
```

```
        rango_y = c(seq(lanz_ultimo[2] - 4, lanz_ultimo[2] - 1), seq(lanz_ultimo[2] +
            1, lanz_ultimo[2] + 4))
    }
}
## Genera 24 combinaciones posibles dentro del rango estipulado
commander_table <- data.frame(posiciones_x = 0, posiciones_y = 0, jugadas_x = 0,
    jugadas_y = 0)
for (i in 1:24) {
    commander_table[i, 1] <- sample(rango_x, 1)
    commander_table[i, 2] <- sample(rango_y, 1)
    commander_table[i, 3] <- commander_table[i, 1]
    commander_table[i, 4] <- commander_table[i, 2]

}
## Consulta con la red
dec_matrix <- predict(nn_22, commander_table[], type = "class")
dif <- vector("integer", length = length(dec_matrix))
dec_matrix <- cbind(commander_table[, 3:4], dec_matrix, dif)

## Calcula la distancia respecto a 1 (buena jugada)
for (i in 1:nrow(dec_matrix)) {
    dec_matrix[i, 4] <- abs(1 - (abs(dec_matrix[i, 3])))
}
## el blanco seleccionado es el que tenga menor diferencia respecto a 1
blanco <- dec_matrix[which.min(dec_matrix$dif), ]
blanco <- as.integer(blanco[1, 1:2])
###### Evitar repetidos, coge el 2º menor. Si este también está, random####
if ((any(sapply(hist_lanzamientos, identical, blanco)))) {
    blanco_2 <- dec_matrix[which.min(dec_matrix$dif != min(dec_matrix$dif)),
        ]
    blanco_2 <- as.integer(blanco_2[1, 1:2])
    if ((any(sapply(hist_lanzamientos, identical, blanco_2)))) {
        blanco <- dec_matrix[sample(nrow(dec_matrix)), 1:2]
        blanco <- as.integer(blanco[1, 1:2])
    } else {
        blanco <- blanco_2
    }

}

##### Corrección de rango #####
if (blanco[1] <= 0) {
    blanco[1] <- 1
} else if (blanco[1] >= 11) {
    blanco[1] <- 10
}

if (blanco[2] <= 0) {
    blanco[2] <- 1
} else if (blanco[2] >= 11) {
    blanco[2] <- 10
}
beep(7)
```

```
    print(blanco)
    return(blanco)
}

alcance_pregunta <- 0
hist_lanzamientos <- vector("list")



#### Ejemplo partida. Esto habría que ponerlo en una sesión, y en otra el
#### sistema de defensa####

lanz_ultimo <- lanzamiento_rand()
hist_lanzamientos[[1]] <- lanz_ultimo
alcance_pregunta <- readline(prompt = "¿Enemigo alcanzado? ")
## CANCELA EL BUCLE ANTES DE VOLVER A TOCAR ARRIBA O CRASHEA. NO IR MUY RÁPIDO
## PULSANDO#######
for (i in 2:60) {
    readline(prompt = "Pulsa cualquier tecla para empezar tu turno ")
    lanz_ultimo <- lanzamiento()
    hist_lanzamientos[[i]] <- lanz_ultimo
    alcance_pregunta <- readline(prompt = "¿Enemigo alcanzado? ")
}
beep(3)
```