Juego Black Jack
Un Análisis de sus Algoritmos

Pontificia Universidad Javeriana, Analisis de Algoritmos *Curso, 1063*

***Abstract*—En este documento se presenta un análisis basado en el juego Blackjack, el cual es uno de los juegos de casino más jugados en el mundo. Se realiza una revisión de los distintos algoritmos realcionados para determinar su complejidad, su comportamiento ante distintas entradas de datos y el funcionamiento interno de los mismos. Esto con el fin de llegar a conclusiones y generar recomendaciones. En base a esto, se formula la problemática y objetivos del trabajo, que definen los aspectos a tratar y la finalidad del mismo.**

***Index Terms*—Blackjack, algoritmo, análisis, complejidad, eficiencia, comportamiento.**

I. Introducción

EL Blackjack es un juego de banca de casino actualmente uno de los más populares alrededor del mundo. Utiliza mazos de 52 cartas y desciende de una familia de juegos de banca de casino conocida como Twenty-One. Es un juego de cartas de comparación en el que cada jugador compite contra el crupier. El precursor inmediato del Blackjack fue la versión en inglés de veintiuno llamada Vingt-Un , un juego de procedencia probablemente española y cuya primer referencia se encuentra en un libro del autor Miguel de Cervantes.[1]

A lo largo de los años, se han discutido cuáles son las mejores estrategias para aumentar la probabilidad de victoria del jugador, a pesar de que las reglas están diseñadas de tal manera que el oponente (el crupier) siempre tenga una ventaja, por lo cual se han generado diversas estrategias con el fin de minimizar las pérdidas. Este juego ha sido estudiado por matemáticos e informáticos hace varias décadas.

Uno de los primeros autores que escribió acerca de las posibles mejores estrategias para jugar fue el matemático Edward O. Thorp, que escribió un libro llamado Beat the Dealer , que incluía gráficos que mostraban la estrategia "básica" óptima describiendo las posibles acciones a realizar que puede ser pedir carta, plantarse, doblar y dividir, según la mano del jugador, la cual puede ser una mano blanda (que tiene un As), una mano dura (todas las demás combinaciones) o un par. También se han desarrollado algoritmos con el fin de determinar cuál es la mejor jugada a realizar a partir de una mano dada. Un ejemplo de esto es el algortimo genético, que utiliza machine learning realizando procesos similares a la reproducción en la naturaleza con el fin de llegar a generaciones de datos cada vez más acertadas que puedan indicar la jugada óptima para un escenario específico.

Este documento estará centrado en inspeccionar y analizar los algoritmos presentes y relacionados con el juego Blackjack, con el objetivo de generar concluisones acerca del comportamiento de los mismos, su complejidad y la manera en que se podrían mejorar aumentando su eficiencia.

Este documento es el proyecto de la materia analisis de algoritmos.
Manuscrito receivido April 18, 2022; revised August 20, 2022.

II. Determinación del Problema

A. Planteamiento del problema

A partir de las reglas del black jack americano se debe inspeccionar y analizar los algoritmos que hacen posible la ejecución del juego y como las decisiones que toma el jugador durante una partida influye en los mismos. Estos algoritmos seran analizados en los aspectos como su complejidad, el mejor caso y el peor caso. También se evaluara si estos pueden disminuir su complejidad con alguna variante de sus algoritmos. El análisis se va a realizar con el fin de determinar el funcionamiento general y específico del juego, refiriendonos al juego en total y a las jugadas específicas que puede tomar el mismo de acuerdo al desarrollo de la partida.

B. Enunciado del problema

Analizar los algoritmos empleados en el juego black jack americano con una baraja. El analisis comprende evaluaciones asintoticas, complegidad algoritmica para el peor y el mejor caso de los algoritmos.

III. Justificación

La justificación de la investigación se basa en el entendimiento del algoritmo presente en el juego Blackjack a partir de los temas aprendidos a lo largo de la materia de Anális de Algoritmos.

A corto plazo esta investigación nos va a dar información sobre la eficiencia del algoritmo de acuerdo a sus tiempos de ejecución en diferentes estados de la partida, para conocer mejor su funcionamiento y el impacto de las de decisiones tomadas por el jugador. A largo plazo, podemos utilizar estos datos para encontrar una mejora del tiempo de ejecucion y optimizacion de dichos algoritmo en caso de que sea posible. Este proceso de investigación y recolección de datos es importante para velar por el ahorro de recursos de computación y realizar análisis a juegos de este estilo, con el fin de brindarle al usuario la mejor experiencia posible. El proyecto es innovador en el ambito del análisis de algoritmos, ya que está encaminado en produndizar el conocimiento sobre el funcionamiento de los algoritmos de un juego popular.

IV. Objetivos de la investigación

A. Objetivo Principal

Analizar los algoritmos del juego Blackjack americano con el fin de entender a profundidad su funcionamiento y explorar posibilidades de predicción y optimización de los mismos.

B. Objetivos Específicos

Comprender los algoritmos utilizados en el juego Black-Jack.

Clasificar los algoritmos de acuerdo a su complejidad algorítmica para el mejor y peor de los casos teniendo en cuenta las decisiones del jugador.

Realizar análisis asintótico a los algoritmos del juego.

Buscar estrategias y observar como estas ayudan a tomar mejores desiciones durante una jugada.

Realizar análisis asintótico a los algoritmos del juego.

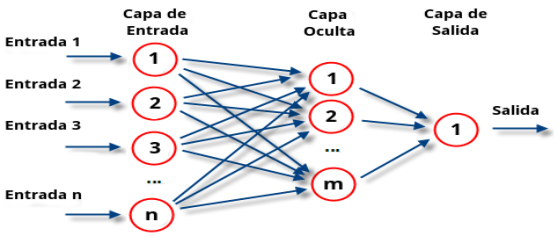
Encontrar posibles mejoras para los algoritmos usados en el juego.

V. Marco Teórico

A. Red neuronal

Una red neuronal se define como un sistema que consiste en una serie de elementos todos interconectados, llamados neuronas, que se organizan en capas que procesan la información utilizando respuestas de estado dinámico a entradas externas.

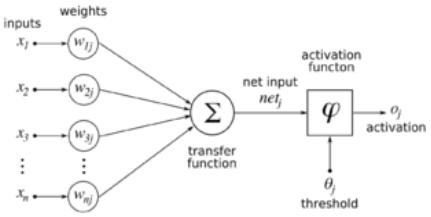
En el contexto de esta estructura, la capa de entrada introduce patrones en la red neuronal que tiene una neurona para cada componente presente en los datos de entrada y se comunica a una o más capas ocultas, se considera capas ocultas a todas las capas de la red exceptuando la de entrada y salida. Es en las capas ocultas donde sucede todo el procesamiento, a través de un sistema de conexiones caracterizado por pesos y sesgos (comúnmente conocidos como W yb)



Con el valor de entrada que recibe la neurona se calcula una suma ponderada agregando también el sesgo y de acuerdo con el resultado y una función de activación preestablecida (las cuales veremos a continuación), se decide la activación o excitación de la neurona. Posteriormente, la neurona transmite la información a otras neuronas conectadas en un proceso llamado "PassFoward". Al final de este proceso, la última capa oculta está vinculada a la capa de salida que tiene una neurona para cada posible salida deseada.

$\sum XiWi + bi$

a = F(z)



Función de activación lineal: Las cuales ya no se usan en el Deep Learning, ya que la salida de las funciones no estará confinada entre ningún rango y la suma de diferentes funciones lineales sigue siendo una función lineal acotando así la activación de la neurona.

Funciones de activación no lineal: Son las usadas en las redes neuronales, como veremos a continuación estas funciones permiten un acotamiento de los datos de salida. Algunos ejemplos son al función sigmoide o tangente hiperbólica.

El paso final del PassForward es evaluar la salida predicha (Yr) contra una salida esperada (Yr). La salida Yr es parte del conjunto de datos de entrenamiento (x, y) donde x es la entrada (como vimos en la sección anterior). La evaluación entre Yp e Yr se realiza a través de una función de coste. Para este ejercicio hemos usado dos funciones de coste:

MSE (error cuadrático medio)

Entropía cruzada binaria.

Llamamos a esta función de coste C y la denotamos de la siguiente manera:

C = cost(Yp,Yr)

Yp= valor que ha predecido nuestra red

Yr=valor real de los datos

Donde el cost puede ser igual a MSE, entropía cruzada o cualquier otra función de coste. Según el valor de C, el modelo sabe cuánto ajustar sus parámetros (Weight y BIAS) para acercarse a la salida esperada y. Esto sucede usando el algoritmo de retropropagación o tambien conocido como Backpropagation.

VI. Estado del arte

El objetivo de la investigación es analizar el algoritmo empleado en el juego Blackjack americano, respodiendo a la pregunta de como funciona el mismo, clasificar los mismos de acuerdo a su complejidad algorítmica, realizar un análisis asintotico y buscar estrategias que ayuden a la toma de decisiones durante una jugada.

Ahora bien para ayudarnos con el tema, podemos guiarnos mediante las investigaciones previamente realizadas por los investigadores van der Genugten en su articulo "Blackjack in Holland Casino's: basic, optimal and winning strategies. Statistica Neerlandica" y Sommerville G en su articulo "Winning Blackjack using Machine Learning"[2]. El artículo de van der Genuten indica de manera detallada las estrategias y el método de optimización utilizado en el juego de cartas Blackjack, siguiendo las reglas del casino Holland en Países Bajos, esta investigación plantea gráficas y ecuaciones para determinar por ejemplo la apuesta media de cada juego y la posibilidad de ganar en determinado número de rondas. Por el otro lado el

articulo de Sommerville toma un enfoque mas técnico, basado en la utilización de nuevas tecnologías y Machine learning para optimizar los algoritmos ya existentes en el juego de Blackjack y utilizarlos para mejorar la toma de decisiones en las diversas situaciones que puede tomar el juego.

Como pudimos ver, el entendimiento de los algoritmos del juego esta bastante avanzado gracias a las investigaciones ya realizadas con el fin de mejorar y entender mejor el desarrollo del juego y las posibles variaciones que puede tener el mismo. Aún asi hay campos de la investigación que siguen sin explorarse, como la optimización directa de los algoritmos utilizando técnicas del análisis de algoritmos.

VII. Modelo de Red Neuronal

A. Primer modelo

EL modelo de la red neuronal densa es de dos capas. La primera capa contiene 4096 neuronas, mientras que la segunda solo 2, para 'h' tomar o 'stay' quedarse. El optimizador de Adam es utilizado, con una pérdida de sparse categorical crossentropy. Los datos de entrenamiento y testeo fueron divididos ambos el 50 porciento aleatoriamente. Se configuraron 10 epocas. EL codigo del modelo es:

```
model = keras.Sequential()
model.add( keras.layers.Dense(4096, input_dim=2) )
model.add( keras.layers.Dense(2, activation=tf.nn.softmax) )
model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
model.fit(train_data, train_tags, epochs=10)
test_loss, test_acc = model.evaluate(test_data, test_tags)
print('Test accuracy:', test_acc)
```

B. Segundo modelo

Este modelo utilizará todas las técnicas anteriores, pero el conjunto de datos ahora incluirá la carta boca arriba del crupier.

Entonces, donde anteriormente usábamos el entero único para los datos, ahora usaremos una tupla de players-hand y dealers-hand respectivamente.

Ejemplo: [18, 13, s]

El conjunto de datos para esto consta de 5323 entradas, ubicadas en data-sets/blackjack.data.2 y data-sets/blackjack.tags.2. La carga de los datos se hace igual que en el modelo 1. Código para generar este conjunto de datos:

```
import Blackjack as bj
bj.gen_data_set( 4000, "test", 2 )
```

La red neuronal utilizó un esquema de capas similar al anterior, con una segunda capa de 16 neuronas. El optimizador era 'nadam' y había 100 épocas. EL código para entrenar esta red es:

```
model = keras.Sequential()
model.add( keras.layers.Dense(16, input_dim=2) )
model.add( keras.layers.Dense(2, activation=tf.nn.softmax) )
model.compile(optimizer='nadam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
model.fit(train_data, train_tags, epochs=100)
test_loss, test_acc = model.evaluate(test_data, test_tags)
print('Test accuracy:', test_acc)
```

Visualizando los resultados de los dos modelos con de acuerdo a la cuenta que llevaba en la jugada de acuerdo a la decesión tomada sobre 's' parar o 'h' para tomar carta se obtuvo lo siguiente:

```
testing model
    4 5 6 7 8 9 0 1 2
5   h h h h h h h h h
6   h h h h h h h h h
7   h h h h h h h h h
8   h h h h h h h h h
9   h h h h h h h h h
10  h h h h h h h h h
11  h h h h h h h h h
12  h h h h h h h h h
13  h h h h h h h h h
14  h h h h h h h h h
15  h h h h h h h h h
16  s s s s h h h h h
17  s s s s s s s s h
18  s s s s s s s s s
19  s s s s s s s s s
20  s s s s s s s s s
21  s s s s s s s s s
```

C. Tercer modelo

```
model = keras.Sequential()
model.add( keras.layers.Dense( 54, input_dim=54 ) )
model.add( keras.layers.Dense( 64, input_dim=26 ) )
model.add( keras.layers.Dense( 128, input_dim=13 ) )
model.add( keras.layers.Dense(2, activation=tf.nn.softmax) )

model.compile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

model.fit(train_data, train_tags, epochs=50)

test_loss, test_acc = model.evaluate(test_data, test_tags)

print('Test accuracy:', test_acc)
```

VIII. Problemas relacionados con los conjuntos de datos

Nivel	Porcentaje
1	51,82
2	41,49
3	41,33

Resultado	Porcentaje
Acertado:	3,42
Estancia:	41,99
Aleatorio:	30,67

El nivel 1 es el modelo más preciso y el modelo se deteriora a medida que obtenemos más información sobre el juego. Los conjuntos de datos claramente necesitan algo de trabajo. El mejor clasificador es solo un 9,17

Problemas potenciales:

-repetir manos

-diferentes conclusiones en el mismo escenario dan como resultado etiquetas opuestas

En un intento por corregir este problema, se escribió un script en /data-sets/preproc.py para preprocesar conjuntos de datos antes de entrenar modelos. El script cumple 2 funciones:

si se encuentran puntos de datos opuestos, se elimina el que tiene el menor número de instancias se eliminan los duplicados El script no se puede ejecutar sin modificarlo. TODO incluye funcionalizar este script e incluirlo en Blackjack.py.

IX. Resultados Obtenidos

A. Primer modelo

El modelo aprendió a tomar o quedarse con la mano actual con un valor menor a 17. Esta estrategia es usada por el dealer.

Para probar el modelo contra un gran un número de simulaciones monte carlo, para esto podemos usar la función testModel() de Blackjack.py. El código es:

```
wins, losses, ties = test_model( "blackjackmodel.1", 10000, True, 1, False )
total = wins + losses + ties
win_percentage = (wins/total)*100.0
loss_percentage = (losses/total)*100.0
tie_percentage = (ties/total)*100.0
print( "Percentage won:  " + str( win_percentage ) )
```

El porcentaje de victorias para 10,000 juegos de este modelo es 52.42 porciento.

B. Segundo modelo

Hay un patrón claro en ambos. Esto confirma que la red neuronal ha empezado a aprender la estrategia del Blackjack.

Para probar el modelo contra un gran número de simulaciones monte carlo, podemos utilizar la función testModel() en Blackjack.py. El código es

```
wins, losses, ties = test_model( "blackjackmodel.2", 10000, True, 2, False )
total = wins + losses + ties
win_percentage = (wins/total)*100.0
loss_percentage = (losses/total)*100.0
tie_percentage = (ties/total)*100.0
print( "Percentage won:  " + str( win_percentage ) )
print( "Percentage lost: " + str( loss_percentage ) )
print( "Percentage tied: " + str( tie_percentage ) )
```

El porcentaje de victorias en 10.000 partidas para este modelo es del 41,49. Curiosamente, esto es menos preciso que el modelo que utilizó menos información sobre el juego. Esto implica que el conjunto de datos es incorrecto, corrupto, etc. Esto será revisado en futuras revisiones.

C. Tercer modelo

Para probar el modelo contra un gran número de simulaciones monte carlo, podemos utilizar la función testModel() en Blackjack.py. El comando es victorias, derrotas, empates = testmodel (”modelo de blackjack.3”, 10000, Verdadero, 3, Falso) total = victorias + derrotas + empates winpercentage = (ganancias/total)*100.0 losspercentage = (pérdidas/total)*100.0 empateporcentaje = (empates/total)*100.0 print(”Porcentaje ganado: ” + str(gananciaporcentaje)) print(”Porcentaje perdido: ” + str(porcentajeperdida)) print(”Porcentaje empatado: ” + str(porcentajeempate))

```
wins, losses, ties = test_model( "blackjackmodel.3", 10000, True, 3, False )
total = wins + losses + ties
win_percentage = (wins/total)*100.0
loss_percentage = (losses/total)*100.0
tie_percentage = (ties/total)*100.0
print( "Percentage won:  " + str( win_percentage ) )
print( "Percentage lost: " + str( loss_percentage ) )
print( "Percentage tied: " + str( tie_percentage ) )
```

El porcentaje de victorias en 10.000 juegos para este modelo es del 41,33. Esto es menos preciso que todos los demás modelos que utilizaron menos información sobre el juego. Esto implica que el conjunto de datos es incorrecto, corrupto, etc. Esto será revisado en revisiones futuras.

References

[1] BB vander Genuten. “Blackjack in Holland Casino’s: basic, optimal and winning strategies. Statistica Neerlandica”. In: 1 (2020). URL: https://search-ebscohost-com.ezproxy.javeriana.edu.co/login.aspx?direct=true&AuthType=ip&db=bth&AN=8615966&lang=es&site=eds-live.

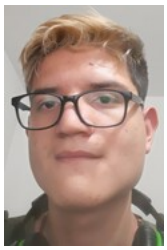
[2] G Sommerville. “Winning Blackjack using Machine Learning”. In: 2 (2019). URL: https://towardsdatascience.com / winning - blackjack - using - machine - learning - 681d924f197c.

X. BIOGRAFÍA

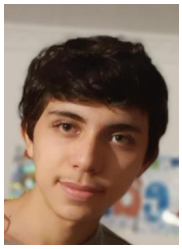
Integrantes de la investigación:

Alejandro Sacristan Leal Miembro IEEE Javeriana en el capitulo RAS. Correo electronico: alejandro-sacristan@javeriana.edu.co

Javier Mauricio Ramírez Portillo Correo electronico: javierm-ramirezp@javeriana.edu.co



Luis Felipe Ayala Urquiza Miembro IEEE Javeriana en el capitulo RAS. Correo electronico: luisfayala@javeriana.edu.co



Andrés José Rodríguez Ortega Correo electronico: rodriguezoa@javeriana.edu.co