Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

REDES NEURONALES APLICADAS A TEORÍA DE JUEGOS

Autora: Eva González de Pablos
Tutores: Óscar Arratia García,
Cesáreo Jesús González Fernández
Junio 2023

Resumen

Las redes neuronales son sistemas de aprendizaje automático relacionados con la Inteligencia Artificial que nos permiten resolver gran cantidad de problemas. Para su aprendizaje necesitan de gran cantidad de datos, es por ello que están relacionadas con muchas técnicas empleadas en estadística sobre el análisis de datos.

La teoría de juegos es de gran importancia en muchos problemas de economía y sirve para entender como se comporta un ser humano en la toma de decisiones cuando un problema se le presenta. Estos últimos años, con el desarrollo de la informática, varios de estos juegos han servido como modelo para desarrollar la inteligencia artificial.

Por estos motivos, una vez elegido un modelo de la Teoría de Juegos y teniendo en cuenta las estrategias que intervienen en ese juego, el objetivo de estudio ha sido aplicar diferentes tipos de redes neuronales para analizar la eficiencia y determinar cuales de ellas nos pueden ofrecer con mayor seguridad la mejor solución del juego estudiado. Para su implementación se ha utilizado el lenguaje de programación Python.

Palabras clave: Redes neuronales, Teoría de Juegos, Juegos bipersonales, Equilibrio de Nash , Inteligencia artificial.

Abstract

Neural networks are machine learning systems related to Artificial Intelligence that allow us to solve a large number of problems. For their learning they need a large amount of data, that is why they are related to many techniques used in statistics on data analysis.

Game theory is of great importance in many economic problems and it serves to understand how a human behaves in making decisions when a problem is presented. In recent years, with the development of computing, several of these games have served as models for develop the artificial intelligence.

For these reasons, once a Game Theory model has been chosen and considering the strategies involved in this game, the objective of the study has been applying different types of neural networks to analyze the efficiency and determine which of them they can offer us with more security the best solution of the studied game. For its implementation, the Python programming language has been used.

Keywords: Neural networks, Game Theory, Two-person games, Nash equilibrium, Artificial intelligence.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La teoría de juegos es una de las ramas de las matemáticas que tiene como objetivo determinar la mejor estrategia para cada jugador teniendo en cuenta que las decisiones de los otros jugadores también afectará al resultado del juego. Es de gran interés en economía, pero este no es su único campo sino que también destacan sus aplicaciones en muchas otras áreas como la biología, política, psicología o informática, entre otras.

Podemos distinguir varios tipos de clasificación de estos juegos de acuerdo a diferentes criterios, como pueden ser: respecto al tipo de las alianzas entre jugadores, en cuyo caso se habla de juegos cooperativos y no cooperativos, en función de la forma en la que se toman las decisiones, clasificándolos en estáticos y dinámicos, o la cantidad de información de la que dispone un jugador, lo que permite establecer los juegos de información completa o incompleta, entre otros criterios. En este trabajo se han elegido para su estudio los juegos bipersonales, es decir, juegos de 2 jugadores y con número de estrategias finito.

Por otro lado, hemos tratado las redes neuronales, las cuales simulan el comportamiento de un cerebro humano y consisten en un conjunto de unidades artificiales que reciben el nombre de neuronas artificiales. Estas se conectan entre sí para transmitir señales, mediante unos pesos o ponderaciones que se van actualizando en función de las entradas que reciben y las señales de salida que producen. La redes neuronales constan generalmente de tres partes: una capa de entrada, una o varias capas ocultas y una capa de salida.

Las redes neuronales están basadas en el aprendizaje por repeticiones teniendo en cuenta si es mejor, o no, usar las diferentes estrategias que intervienen para llegar a obtener el mejor resultado posible. Es por ello que optimizar la convergencia es muy importante, del mismo modo que lo es tener un porcentaje elevado de aciertos a la hora de predecir el resultado final.

Nuestro objetivo en este trabajo es comprobar cuáles son las redes neuronales que nos ofrecen un mejor resultado para un tipo específico de juego y en un menor tiempo, es decir mayor

velocidad de convergencia, para posteriormente poder extraer conclusiones.

En este trabajo, en el Capítulo 2, describiremos los conceptos de la teoría de juegos necesarios para comprender las técnicas aplicadas, así como, en el Capítulo 3, los conceptos relacionados con las redes neuronales, los diferentes tipos que podemos encontrar y su funcionamiento. En el Capítulo 4, terminaremos uniendo ambos temas, bajo una implementación desarrollada en Python, cuyo código se incluirá junto a esta memoria, en el que aplicamos las diferentes redes neuronales elegidas sobre nuestro juego en cuestión y obtenemos una serie de tablas y resultados gráficos que nos permitirán llegar a obtener conclusiones respecto al objetivo de nuestro estudio.

Capítulo 2

Teoría de juegos

2.1. Marco histórico

Los primeros trabajos más relevantes que constituyen las bases de lo que ahora se conoce como teoría de juegos fueron elaborados en 1944 por John von Neumann quien, en colaboración, con Oskar Morgenstern publicó el primer estudio [5].

No obstante, existen otras obras anteriores que merecen ser mencionadas pues ya hacían referencia a conceptos importantes para comprender la teoría de juegos, como [6], escrita, en 1704 por Leibniz. En ella se hablaba de "una nueva clase de lógica para predecir los juegos de azar". Leibniz afirmaba que la mente humana "se despliega más minuciosamente en los juegos que en actividades más serias". También fue relevante el estudio publicado en 1713 [2], cuyo autor fue Pierre-Rémond, trata lo que hoy se conoce como solución minimax, una estrategia que consiste en asegurar la máxima ganancia en el peor caso posible para cada jugador, que coincide con el equilibrio de Nash. El estudio se continuó desarrollando por Montomort, Bernoulli y Waldegrave, quienes publicaron algún trabajo más al respecto.

Otro autor importante a destacar y ya en años más recientes, hacia 1950, fue Nash, quien aportó unos conceptos muy importantes en el ámbito de la teoría de juegos. Años más tarde otros autores como Selten y Harsanyi completaron estas ideas, haciendo posible la aplicación de la teoría de juegos en la economía.

Todos estos investigadores analizan el comportamiento de los individuos dependiendo de las posibles estrategias a seguir.

2.2. Tipos de juegos

En función de las alianzas que se establecen entre los jugadores podemos distinguir dos tipos de juegos: cooperativos, aquellos en los que los jugadores forman equipo y se ponen de acuerdo respecto de la decisión a tomar, y no cooperativos, en los cuales los jugadores no toman las decisiones comúnmente entre ellos.

También podemos diferenciar los juegos estáticos, en los que los jugadores toman las decisiones sin saber qué decidieron los otros jugadores, y juegos dinámicos en los que, por el contrario, sí se conocen total o parcialmente las decisiones de otros jugadores y afectan a sus decisiones.

Otra clasificación atiende a la cantidad de información disponible: los juegos se denominan de información completa si todos los jugadores conocen lo que ocurrirá, tanto con sus decisiones como con las de los demás, y de información incompleta cuando alguno de los jugadores no conoce en su totalidad las consecuencias que tendrán las acciones.

Además, podemos distinguir diferentes tipos de juego en función de los pagos que recibe un jugador. Aquí se encuentran los juegos de suma cero, en los cuales se incluyen aquellos donde lo que gana un jugador es exactamente lo que pierde otro, si esto no sucede se clasifican como de suma no cero. Hay casos en los que los juegos no son exactamente de suma cero, pero se pueden convertir fácilmente transformando las ganancias que obtienen los jugadores sumando o restando una determinada cantidad, estos juegos se denominan de suma constante.

Si tenemos en cuenta el número de estrategias tenemos los juegos finitos, aquellos con un número finito de estrategias, y los juegos infinitos, cuando las posibles estrategias no son finitas.

2.3. Elementos de un juego

Un juego está compuesto por una serie de elementos los cuales definimos a continuación:

1. Jugadores: Son los actores del juego, dos o más, que toman las decisiones del juego en busca de un objetivo, maximizar su ganancia. En este trabajo se supondrán jugadores racionales [1].

2. Acción: Es cada una de las decisiones que puede tomar un jugador durante el juego cada vez que puede participar en él. Estas pueden ser simples o complejas.

3. Estrategia: Es el conjunto completo de acciones con las que participará un jugador a lo largo del juego. Un perfil de estrategias es el conjunto de las estrategias usadas por el total de los jugadores del juego.

4. Resultado: Son los diferentes estados en los que termina un juego.

5. Solución del juego: Es un concepto de difícil definición [1] .De forma intuitiva la solución de un juego está constituida por un perfil de estrategias que, a priori, será la respuesta racional de los jugadores que intervienen.

6. Pagos: es la utilidad esperada que recibirá un jugador al terminar el juego. Por lo general son numéricos y sirven para cuantificar las consecuencias que tendrán las acciones de cada jugador a lo largo del juego.

2.4. Formas de un juego

Son las posibles representaciones de un juego, describiendo los jugadores, las posibles estrategias a seguir y los pagos correspondientes de cada jugador. Encontramos la forma estratégica o normal y la forma extensiva

2.4.1. Forma estratégica o normal

Esta forma se basa en representar el juego de forma matricial, incluyendo las estrategias de los jugadores con los correspondientes pagos que estos obtendrían, separados por comas para distinguir a qué jugador pertenece. En la Figura 2.1 aparece un ejemplo.

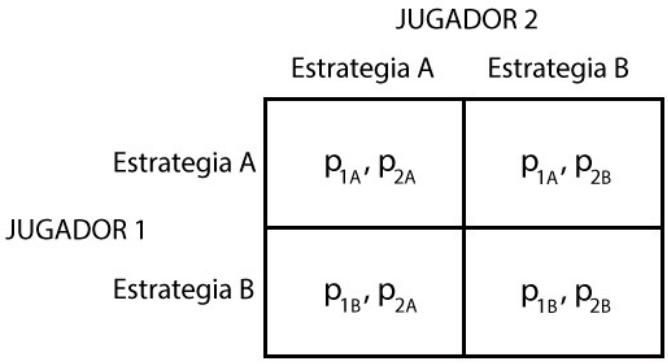

Figura 2.1: Representación en forma estratégica
Fuente: [19]

2.4.2. Forma extensiva

En la forma extensiva el juego está descrito mediante árboles y se basa en representarlo de forma secuencial. Los nodos representan los jugadores, las aristas las acciones que toma cada jugador y los nodos terminales representan los posibles resultados. Un ejemplo sería el grafo 2.2

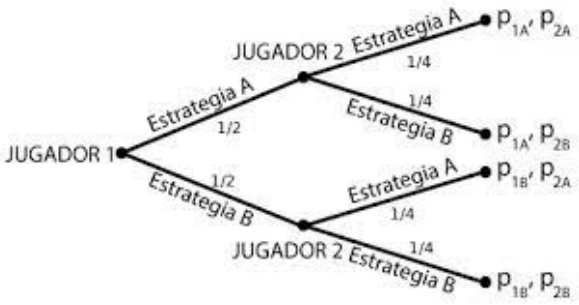

Figura 2.2: Representación en forma extensiva
Fuente: [19]

2.5. Equilibrio de Nash (EN)

El equilibrio de Nash es un concepto muy importante en la solución de un juego. Un equilibrio de Nash es un perfil de estrategias del juego en el que ningún jugador puede aumentar sus ganancias por un cambio unilateral de estrategia. Si un jugador pudiera aumentar sus ganancias sería difícil terminar considerando esa solución como la final del juego ya que uno de los jugadores cambiaría la elección de su estrategia. El EN no será el mejor resultado posible para cada jugador sino aquel en el que todo los jugadores obtendrán el mejor beneficio posible de acuerdo a las posibles estrategias. Puede existir uno, varios o ningún equilibrio de Nash en un juego.

Para encontrar los equilibrios de Nash en un juego podemos usar varios métodos.

2.5.1. Método de los pagos subrayados

Es una manera sencilla y visual de hallar los equilibrios de Nash en un juego utilizando las matrices de pagos. Consiste en subrayar el pago máximo que podría conseguir un jugador si se jugara una determinada estrategia por parte de otro jugador. Los equilibrio de Nash estarán situados en los cuadros de la matriz donde se encuentran todos los valores subrayados.

		Jugador 2		
		I	C	D
Jugador 1	A	$\underline{3}, \underline{3}$	$\underline{4}, 2$	$1, 1$
	M	$2, 4$	$3, \underline{5}$	$\underline{4}, 0$
	B	$1, 0$	$2, 1$	$0, \underline{3}$

Tabla 2.1: Matriz de pagos subrayados

La Tabla 2.1 muestra un ejemplo del método de pagos subrayados. Explicaremos, a modo de ejemplo cómo se subraya un elemento: el jugador 2 juega la estrategia C, entonces el jugador 1 podría ganar 4, 3 o 2. De ellos el máximo es 4 y, por tanto, se subraya. Análogamente para el resto de estrategias.

Podemos observar que el único elemento que está doblemente subrayado es la estrategia A por parte del jugador 1 y la estrategia I por parte del jugador 2. Es, por tanto, el equilibrio de Nash para esta matriz de pagos.

2.5.2. Método de las desviaciones deseadas

En este método se señala hacia que estrategia se desviaría el juego porque uno de los jugadores conseguiría una mayor ganancia. Cuando estando en un par de estrategias no hay ninguna posible desviación hacia otro par de estrategias ese será un equilibrio de Nash.

En el ejemplo de la Figura 2.3 podemos ver que el único par de estrategias que no cuenta con ninguna flecha de salida, y por lo tanto es un equilibrio de Nash, es el mismo que obtuvimos con el método anterior, (A, I). Nótese, por ejemplo, que el par (M, I) no es EN, ya que al jugador 1 le interesaría ir hacia la estrategia A o al jugador 2 hacia la estrategia C.

2.5.3. Estrategia maximín

Para juegos de dos jugadores donde lo que gana uno de ellos es lo que pierde el otro jugador, lo que se denomina como juego de suma cero, se dispone de otra estrategia para el calculo de los equilibrios de Nash. En este trabajo no se ha utilizado esta estrategia como método de resolución ya que las matrices que hemos decidido utilizar no representan juegos de suma cero, pero es necesario mencionarla debida a su relevancia.

En primer lugar, comenzaremos explicando los conceptos necesarios para poder entender esta forma de encontrar los equilibrios de Nash.

		Jugador J2				
		I		C		D
Jugador J1	A	3, 3	⇐	4, 2	⇐	1, 1
		⇑		⇑		⇓
	M	2, 4	⇒	3, 5	⇐	4, 0
		⇑		⇑		⇑
	B	1, 0	⇒	2, 1	⇒	0, 3

Figura 2.3: Matriz desviaciones deseadas
Fuente: [1]

Consideremos el juego J= $\{S_1,S_2,u_1,u_2\}$, donde S_1 son las estrategias del jugador 1, S_2 las estrategias del jugador 2, u_1 son los pagos para el jugador 1 y u_2 para el jugador 2. Denotaremos por M_1 a la matriz de pagos del jugador 1 cuyos elementos serán $u_1(s_1^i, s_2^j)$.

Definimos el valor maximin como:

$$\max_{s_1^i \in S_1} \{ \min_{s_2^j \in S_2} u_1(s_1^i, s_2^j) \}$$

es decir, el máximo de los mínimos de las filas.
Por otro lado, se define el valor minimax como:

$$\min_{s_2^j \in S_2} \{ \max_{s_1^i \in S_1} u_1(s_1^i, s_2^j) \}$$

es decir, el mínimo de los máximo de las columnas.

Los equilibrios de Nash para juegos bipersonales finitos de suma cero están directamente conectados con el concepto maximin cuando se consideran estrategias mixtas en el juego. (Ver [1])

Para ilustrar el método utilizaremos el ejemplo de la Tabla 2.2.

Para la primera fila los valores correspondientes para el J1, en rojo, son 9, 1 y 2, de ellos el mínimo es 1. Para la segunda fila entre 6, 5 y 4, el mínimo es 4 y para la tercera, entre 7, 8 y 3, el mínimo es 3. Ahora, tenemos lo valores 1, 4 y 3, de ellos nos quedamos con el máximo, en este caso el 4.

Para el J2 tenemos que fijarnos en las columnas y los números en color azul. Para la primera columna entre 1, 4 y 3 el valor mínimo es 1. Para la segunda columna entre 9, 5 y 2 el mínimo

		J2		
		A	B	C
J1	A	9, 1	1, 9	2, 8
	B	6, 4	5, 5	4, 6
	C	7, 3	8, 2	3, 7

Tabla 2.2: Estrategia maximin

es 2. Para la tercera columna 8, 6 y 7 el mínimo es 6. Ahora tenemos 1, 2 y 6 de ellos el máximo es 6.

Encontramos nuestro equilibrio de Nash en el punto (4, 6) que corresponde con la estrategia B del J1 y C del J2.

2.6. Ejemplos de juegos

En esta sección ilustraremos brevemente algunos de los modelos simplificados de juegos que sirven como base de problemas de economía y señalaremos en cada uno de ellos la matriz de pagos y el equilibrio de Nash.

2.6.1. El dilema del prisionero

El dilema del prisionero es el problema más clásico de la teoría de juegos, en él, dos sospechosos de un robo se encuentran cada uno ante dos posibles situaciones: confesar o callar. Se enuncia de la siguiente forma [1]:

"Dos delincuentes habituales son apresados cuando acaban de cometer un delito grave. No hay prueba clara contra ellos, pero sí indicios fuertes de dicho delito y además hay pruebas de un delito menor. Son interrogados simultáneamente en habitaciones separadas. Ambos saben que si los dos se callan serán absueltos del delito principal por falta de pruebas, pero condenados por el delito menor (1 año de cárcel), que si ambos confiesan, serán condenados por el principal pero se les rebajará un poco la pena por confesar (4 años), y finalmente, que si sólo uno confiesa, él se librará de la pena y al otro «se le caerá el pelo» (5 años)."

Podemos representarlo mediante la matriz de pagos Tabla 2.3:

		Preso 2	
		Callar	Confesar
Preso 1	Callar	$-1, \quad -1$	$-5, \quad 0$
	Confesar	$0, \quad -5$	$-4, \quad -4$

Tabla 2.3: Dilema del prisionero, matriz de pagos

Escalada de forma estándar la matriz se puede reescribir de tal modo que se eliminan los valores negativos, según se ve en la Tabla 2.4.

		Preso 2	
		Callar	Confesar
Preso 1	Callar	$4, \quad 4$	$0, \quad 5$
	Confesar	$5, \quad 0$	$1, \quad 1$

Tabla 2.4: Dilema del prisionero, matriz de pagos escalada

Como podemos ver, en este juego hay cuatro posibles combinaciones de estrategias o perfiles del juego.

Utilizando una de las técnicas para la obtención de equilibrios de Nash, por ejemplo el método de pagos subrayados, tendríamos: Para el preso 1, si el preso 2 elige callar, la mejor opción sería confesar puesto que le ofrece un beneficio de 5 frente a 4 que le proporcionaría callar, en caso de que el preso 2 confesara, el preso 1 elegiría confesar también, puesto que su ganancia sería 1, en lugar de 0 si se quedara callado. Para el preso 2 ocurre algo similar, si el preso 1 elige callar la mejor opción sería confesar y si el preso 1 elige confesar, el preso 2 confesaría también.

Si dibujamos esto sobre nuestra matriz de pagos obtenemos la Tabla 2.5, donde podemos ver cómo el único perfil del juego doblemente subrayado corresponde a confesar por parte de ambos presos.

		Preso 2	
		Callar	Confesar
Preso 1	Callar	$4, \quad 4$	$0, \quad \underline{5}$
	Confesar	$\underline{5}, \quad 0$	$\underline{1}, \quad \underline{1}$

Tabla 2.5: Matriz de pagos subrayados - Dilema del prisionero

Como conclusión, de este juego podemos extraer que cuando dos jugadores están buscando el mayor beneficio individual, obtendrán un resultado peor que si lo hubieran pensado de forma grupal, ya que en nuestro equilibrio de Nash, ambos presos obtendrán una pena de cárcel de 4

años, respecto a la opción callar por ambos presos, en la cual únicamente hubieran obtenido 1 año de cárcel cada uno. Las otras 2 opciones reafirman nuestra conclusión, ya que si uno de los dos no confiesa y el otro lo hace, estaría obteniendo 5 años de cárcel.

2.6.2. La guerra de los sexos

Una pareja tiene que decidir qué van a hacer después del trabajo, ir al cine o al fútbol, pero el problema es que ambos aun no están de acuerdo sobre cuál de las dos opciones elegir y tampoco tendrán tiempo para hablarlo después del trabajo, por lo que cada uno, sin saber qué hará el otro, debe elegir dónde ir directamente. Ellos prefieren estar juntos a estar separados, pero cada uno de ellos tiene preferencia por una de las opciones: el jugador 1 prefiere el fútbol, y la jugadora 2 el cine.

Podemos verlo en la matriz de pagos representada en la Tabla 2.6.

		Jugadora 2	
		Cine	Fútbol
Jugador 1	Cine	1, 2	0, 0
	Fútbol	0, 0	2, 1

Tabla 2.6: Matriz de pagos guerra de los sexos

En este juego también observamos cómo hay 4 perfiles de juego: cine, cine; cine, fútbol; fútbol, cine y fútbol, fútbol.

Si comprobamos los equilibrios de Nash podemos ver cómo, en este juego, tenemos 2 equilibrios, cine, cine y fútbol, fútbol, que representado mediante el método de pagos subrayados sería la Tabla 2.7:

		Jugadora 2	
		Cine	Fútbol
Jugador 1	Cine	$\underline{1}$, $\underline{2}$	0, 0
	Fútbol	0, 0	$\underline{2}$, $\underline{1}$

Tabla 2.7: Matriz de pagos subrayados guerra de los sexos

Solo conseguirán un pago positivo si ambos coinciden a la hora de elegir el sitio.

2.6.3. Modelo Halcón Paloma

Enunciamos el modelo de la siguiente forma: dos seres vivos pueden comportarse de dos formas, agresiva, halcón, o pacífica, paloma, a la hora de conseguir un objetivo. Estos pueden ambos decidir pelearse para conseguirlo, cooperar ambos, o uno de ellos pacífico, sin obtener nada, y el otro agresivo obteniendo todo.

C es el coste que supone que ambos se comporten agresivamente.

Lo representamos mediante la matriz de pagos Tabla 2.8:

		Jugador 2	
		Paloma	Halcón
Jugador 1	Paloma	$V/2,\quad V/2$	$0,\qquad V$
	Halcón	$V,\qquad 0$	$V/2-C,\quad V/2-C$

(donde $V > 0, C > 0$)

Tabla 2.8: Matriz de pagos - Halcón Paloma

Si buscamos el equilibrio de Nash lo obtenemos en el perfil de ambos comportándose de forma agresiva Tabla 2.9.

		Jugador 2	
		Paloma	Halcón
Jugador 1	Paloma	$V/2,\quad V/2$	$0,\qquad V$
	Halcón	$V,\qquad 0$	$\underline{V/2-C},\quad \underline{V/2-C}$

(donde $V > 0, C > 0$)

Tabla 2.9: Matriz de pagos subrayados - Halcón Paloma

2.6.4. La caza del ciervo

El enunciado del juego es el siguiente: dos personas van de caza juntas a un coto privado y cada una de ellas tiene dos opciones, permanecer en el puesto con el objetivo de cazar un ciervo, o intentar cazar el ciervo y las liebres al mismo tiempo. Solo podrán cazar al ciervo si ambos cazadores se mantienen en su puesto, sin pensar en las liebres, pero si uno de ellos decide cazar las liebres no logran cazar al ciervo. Ambos prefieren un ciervo a las liebres pero al mismo tiempo las liebres a no llevarse nada de vuelta a casa.

La matriz de pagos que representa a este juego es la Tabla 2.10.

		Jugador 2	
		Cooperar	Buscar liebre
Jugador 1	Cooperar	V, V	0, $2W$
	Buscar liebre	$2W$, 0	W, W

(donde $V > 2W, W > 0$)

Tabla 2.10: Matriz de pagos - Caza del ciervo

Utilizando el método pagos subrayados para encontrar el equilibrio de Nash tenemos la Tabla 2.11.

		Jugador 2	
		Cooperar	Buscar liebre
Jugador 1	Cooperar	$\underline{V}$, $\underline{V}$	0, $2W$
	Buscar liebre	$2W$, 0	$\underline{W}$, $\underline{W}$

(donde $V > 2W, W > 0$)

Tabla 2.11: Matriz de pagos subrayados - Caza del ciervo

A la vista de esta matriz podemos decir que hay dos equilibrios de Nash, que corresponden a cooperar, cooperar y a buscar liebre, buscar liebre.

Este juego vuelve a resaltar la importancia de cooperar para conseguir el mayor beneficio.

Capítulo 3

Redes neuronales

La inteligencia artificial, IA, trata de conseguir respuestas a nuestros a problemas a partir de soluciones y métodos tecnológicos cuyos comportamientos tratan de simular el del cerebro humano.

Muchas de las técnicas que se emplean en la inteligencia artificial y el aprendizaje automático son parte de la ciencia de datos y de las redes neuronales artificiales. Para el correcto funcionamiento de estas técnicas, es necesario contar con una gran cantidad de datos de los que se pueda extraer la información necesaria que sirva de entrenamiento para estos modelos de aprendizaje y nos ayude a conseguir grandes resultados.

3.1. Breve historia de las redes neuronales

La primera neurona artificial fue desarrollada en 1943 por Warren Sturgis McCulloch y Walter Harry Pitts: la conocida como "Neurona de McCulloch-Pitts", estaba basada en las matemáticas y en la lógica de Umbral. Este descubrimiento provocó que las redes neuronales pudieran desarrollarse siguiendo dos enfoques, uno basado en el cerebro humano y otro en la aplicación de la inteligencia artificial.

Posteriormente, en 1949, Donald Hebb, en su obra The Organization of Behavior[4], destacó la importancia de la mejora del aprendizaje cuando dos nervios actúan al mismo tiempo.

Cerca de la década de los 50 fue posible por primera vez hacer la simulación de una red neuronal, aunque su implementación falló. Posteriormente, en 1956, un estudio [3] resuelve los problemas y abre el camino a la investigación en inteligencia artificial y el procesamiento neuronal, pues introducía cómo los ordenadores podían simular el aprendizaje.

En 1959 aparecieron los conceptos “ADALINE” y “MADALINE”. El primero de ellos era capaz de reconocer patrones de bit, mientras que el segundo fue la primera red neuronal en un problema del mundo real.

En 1986, surgieron las redes backpropagation que proporcionaban un resultado más preciso, pero más lento. Actualmente se trata de conseguir que las redes neuronales sean rápidas con la mejora en el desarrollo del hardware.

En la Figura 3.1 tenemos una línea temporal sobre el desarrollo de las redes neuronales:

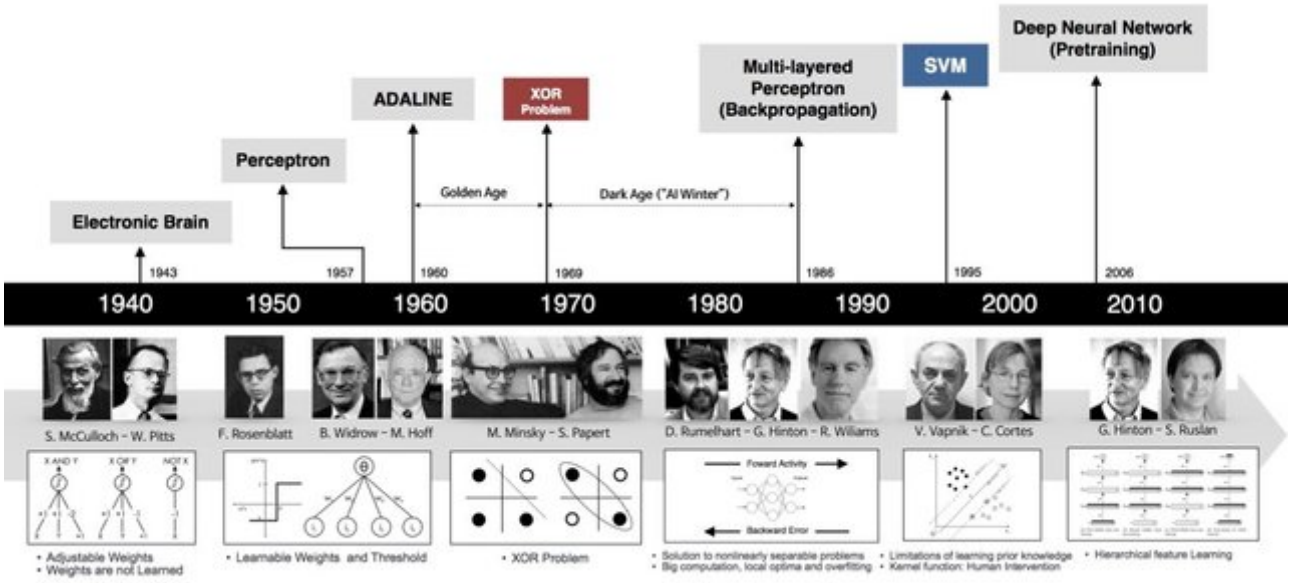

Figura 3.1: Marco histórico de las redes neuronales
Fuente: [21]

3.2. ¿Qué es una red neuronal?

Como ya hemos indicado con anterioridad, una red neuronal es un conjunto de elementos artificiales, con una estructura muy similar a nuestras neuronas, interconectadas entre sí, para lograr, ante unos estímulos de entrada, una salida determinada. Dichos elementos se denominan neuronas artificiales y se encuentran distribuidos en una serie de capas: una capa de entrada, una o varias capas ocultas y una capa de salida.

La Figura 3.2 muestra un esquema de una red neuronal, con una capa de entrada, una capa oculta y una capa de salida.

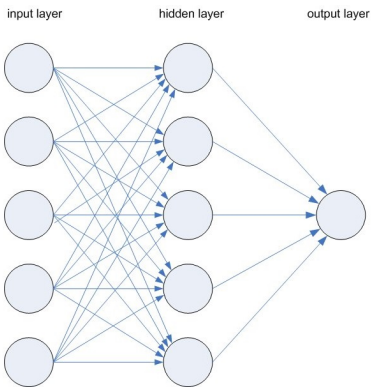

Figura 3.2: Estructura básica de una red neuronal
Fuente: [28]

La característica principal de las neuronas artificiales es el auto-aprendizaje basado en la experiencia, con el objetivo de reducir progresivamente el número de errores que se cometen.

Las neuronas artificiales están formadas por:

- Conjunto de entradas: parte por donde se recibe la información del exterior o de otras neuronas. Cada neurona tiene un peso de entrada que se va modificando para conseguir que las salidas se ajusten a la verdadera salida y así reducir el error.
- Funciones de activación, propagación o transferencia.
- Salida de la neurona: resultado de esa neurona tras el procesamiento de los estímulos recibidos, ya sea la clasificación final o la información que va a recibir otra neurona.

Encontramos una representación en la Figura 3.3:

3.3. Tipos de redes neuronales

Podemos clasificar las redes neuronales en función de dos aspectos: según la estructura de la red y según el tipo de aprendizaje.

3.3.1. Redes neuronales según la estructura de la red

La clasificación de las redes neuronales en función de su estructura es la siguiente:

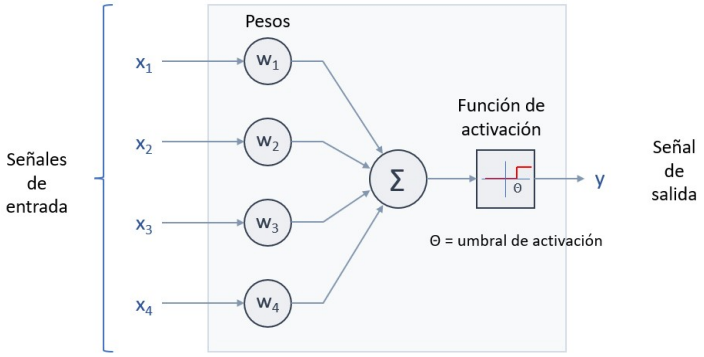

Figura 3.3: Esquema de una neurona artificial
Fuente: [22]

- Monocapa: Es la red neuronal más simple que existe, está formada por una única capa de neuronas que procesa las entradas dando lugar a unos valores de salida, además las neuronas no se conectan entre ellas. Figura 3.4.

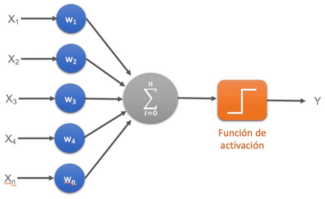

Figura 3.4: Estructura monocapa
Fuente: [25]

- Multicapa: Es la red neuronal más utilizada. Cuenta con una capa de entrada, varias capas intermedias, las llamadas capas ocultas, que procesan la información recibida por las neuronas de la capa de entrada, y el resultado obtenido tras el procesamiento lo transmiten a la capa de salida que dará lugar al resultado final. Las neuronas se conectan con todas las neuronas de la siguiente capa, pero no con las que se encuentran en la misma capa. Figura 3.5.

- Convolucional (CNN): Es la misma idea de estructura que las redes multicapa, pero las neuronas de una misma capa solo se conectan con una parte de las neuronas de la siguiente capa. Tampoco se conectan las neuronas de una misma capa, lo cual consigue que se reduzca el número de neuronas necesarias disminuyendo la complejidad computacional. Figura 3.6.

- Concurrentes: Todas las neuronas se encuentran interconectadas entre ellas en lugar de

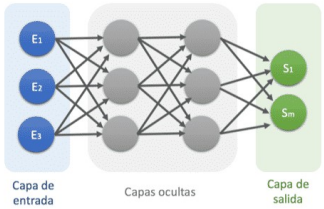

Figura 3.5: Estructura multicapa
Fuente: [25]

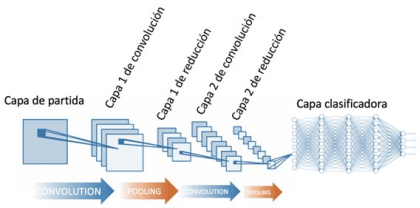

Figura 3.6: Estructura convolucional
Fuente: [25]

formar capas permitiendo crear ciclos, lo cual produce que la red tenga memoria, pues la red aprende de las iteraciones previas. Figura3.7.

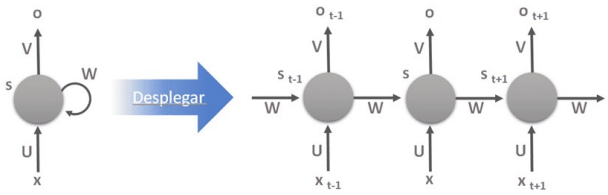

Figura 3.7: Estructura concurrente
Fuente: [25]

- Radiales: La función de salida es una combinación de las funciones de las neuronas que se basan en calcular la distancia a un punto denominado centro. Este tipo de estructura no tiene mínimos locales. Figura 3.8.

3.3.2. Redes neuronales según el tipo de aprendizaje

- Supervisado: En una primera fase de entrenamiento se le suministra a la red una serie de entradas junto con el resultado correspondiente y la red ajusta los pesos de las neuronas

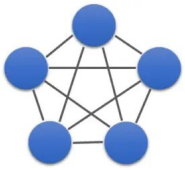

Figura 3.8: Estructura radial
Fuente: [25]

para minimizar el error y llegar a conseguir la mayor cantidad de acierto posible. Puede haber influencia externa a la red que modifique estos pesos para variar el error, lo que se conoce como corrección de error (mínimo error cuadrático (LMS) o backpropagation). Por otra parte, para el aprendizaje de la red, se pueden utilizar técnicas de aprendizaje estocástico que consisten en realizar cambios aleatorios en los pesos para minimizar el error.

- No supervisado: En este caso los pesos se ajustan en función de las semejanzas entre las diferentes entradas, puesto que no se proporciona el resultado que debería dar cada entrada.
- Por refuerzo: Similar al aprendizaje supervisado, pero en lugar de proporcionar el resultado esperado para una serie de entradas, se indica si la salida obtenida es la correcta o no y esto provoca que se vayan ajustando los pesos.

3.4. Ejemplos de redes neuronales

- Perceptrón: Esta red neuronal pertenece a las redes neuronales monocapas; está compuesta únicamente por una capa de entrada y una capa de salida, sin capas intermedias. Lo único que realiza la capa de entrada es transmitir la información recibida a las neuronas de la capa de salida, sin procesar esa información.

  Según el tipo de aprendizaje se clasifica dentro del aprendizaje supervisado.

  Suele funcionar bien cuando el problema es linealmente separable.

- Adaline: Son redes neuronales monocapa al igual que el perceptrón, cuyo aprendizaje también es aprendizaje supervisado, pero, a diferencia del perceptrón, minimizan el error cuadrático medio, utilizando el algoritmo de descenso del gradiente para ajustar los pesos, también conocido como regla delta, lo cual produce una mejora en el aprendizaje.

  Al igual que el perceptrón su limitación consiste en que solo funcionan para problemas linealmente separables.

Para su análisis en el capítulo siguiente, se usará una librería de Python que implementa este clasificador utilizando técnicas de descenso de gradiente estocástico, por ello en los resultados nos referiremos a este clasificador como SGD.

- Perceptrón Multicapa: Esta red neuronal pertenece al grupo de redes neuronales multicapa, es decir, consta de capas ocultas entre la capa de entrada y la capa de salida. Esto nos permite resolver problemas más complejos que no son linealmente separables, es decir, los datos no pueden separarse mediante una línea recta.

  El algoritmo que utiliza como entrenamiento es propagación hacia atrás (regla delta), ajustando los pesos para minimizar el error de predicción. Esto cuenta con problemas de mínimos locales que hacen que el algoritmo se detenga aunque no haya alcanzado la convergencia deseada.

  Es uno de los modelos de red neuronal más utilizado debido a su capacidad para resolver problemas, pero es computacionalmente costoso y requiere un gran número de datos para que el entrenamiento de la red sea correcto.

  En relación con el tipo de aprendizaje, se clasifica dentro del aprendizaje supervisado.

3.5. Aplicaciones de las redes neuronales

Las redes neuronales tienen multitud de aplicaciones y más aun ahora con el desarrollo de la inteligencia artificial. Algunos ejemplos son los siguientes:

- Reconocimiento de voz
- Vehículos de conducción autónoma
- Robótica
- Clasificación
- Regresión
- Procesamiento de datos
- Diagnóstico y predicción
- Reconocimiento visual

Capítulo 4

Redes neuronales aplicadas a teoría de juegos

4.1. Introducción

En este capítulo lo que vamos a desarrollar es la utilización de algunas redes neuronales y una serie de clasificadores de aprendizaje automático supervisado para comprobar su comportamiento en un tipo específico de juego de teoría de juegos.

Lo primero que hicimos fue decidir en qué tipo de juego nos íbamos a centrar, los juegos bipersonales de tres estrategias por jugador, con un objetivo, saber si las redes neuronales elegidas o los clasificadores eran capaces de predecir correctamente los equilibrios de Nash dada una matriz de pagos.

El primer paso fue desarrollar una función que nos permitiera localizar los equilibrios de Nash.

Anteriormente hemos hablado de tres formas de encontrar los equilibrios de Nash, mediante el método de pagos subrayados, desviaciones deseadas y estrategia maximin, pero solo dos de ellos podían ser utilizados en nuestro problema, ya que nuestro juego no es de suma cero, condición necesaria para aplicar el método maximin. Por ello las funciones que desarrollamos fueron el método de pagos subrayados y el método de las desviaciones deseadas.

Debido a la flexibilidad que nos proporcionaba la programación del método de los pagos subrayados frente al de las desviaciones deseadas, a la hora de encontrar los equilibrios para una matriz dada continuamos utilizando la función que hemos construido usando el método de pagos subrayados.

4.2. Algoritmos utilizados para la clasificación y predicción

Para valorar el funcionamiento a la hora de predecir el mejor resultado del juego y, por lo tanto, hallar los equilibrios de Nash hemos utilizado: redes neuronales, algoritmos de aprendizaje automático y regresión logística.

Las redes neuronales que hemos utilizado, ya explicadas en el capítulo de redes neuronales, han sido:

- Perceptrón
- Adaline
- Perceptrón multicapa

Los algoritmos de aprendizaje automático que hemos utilizado son:

- K-Vecinos más próximos: Es un algoritmo de aprendizaje automático que se basa en clasificar un dato en función de la clasificación de un grupo de k vecinos que se encuentren más próximos al dato según una determinada medida, como puede ser la distancia euclídea.

  La distancia euclídea es la más utilizada en la práctica y mide la distancia de un dato a otro en línea recta . También encontramos la distancia de Manhattan que se basa en el valor absoluto entre dos puntos, entre otras distancias. Esto lo podemos ver en la Figura 4.1
  Tiene una serie de ventajas como ser simple y fácil de implementar, es fácilmente adaptable a la introducción de nuevos datos y la fase de entrenamiento es rápida. Sin embargo, también tiene desventajas ya que hay que tener cuidado en la elección de un k adecuado, la fase de clasificación es lenta, ocupa mucho espacio en memoria pues almacena todos los datos, no funciona bien con datos de alta dimensión, lo que se conoce como maldición de la dimensionalidad, lo cual fomenta el sobreajuste.

- Support Vector Machine (SVM): Este algoritmo trata de encontrar un hiperplano que separe los datos de manera que el margen entre las dos clases sea el más amplio posible 4.2.

  Este algoritmo es especialmente útil pues no es necesario que los problemas sean linealmente separables para que encuentre la solución, ya que es capaz de hacer una transformación no lineal del espacio de entrada y dar lugar a un espacio linealmente separable, esto se hace mediante unas funciones llamadas Kernel, que ahorran todo el costo de tener que transformar los datos. Estas funciones pueden ser de base radial (RBF) o gaussiana, lineal, polinómica y sigmoide 4.3.

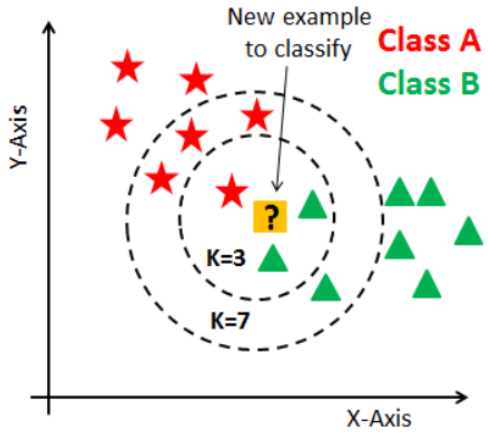

Figura 4.1: Ejemplo de clasificación usando K-vecinos más próximos
Fuente: [20]

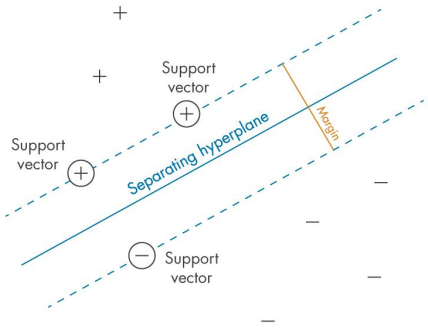

Figura 4.2: Esquema hiperplano en SVM
Fuente: [15]

Hay una serie de características que hacen que SVM sean muy populares:

- Logran un buen rendimiento en clasificación y regresión .
- Son fácilmente transformables para problemas multiclase combinando clasificadores binarios.
- La transformación mediante funciones Kernel hace que sean muy útiles en problemas no lineales.
- Los datos realmente importantes son solamente aquellos que se han seleccionado como puntos soporte, lo cual hace que tengan una representación compacta.

- Regresión logística: Este es uno de los algoritmos de clasificación más utilizados para la clasificación de dos clases, fácil de implementar [24]. Estima el valor de una variable respuesta en función de una o más variables predictoras.

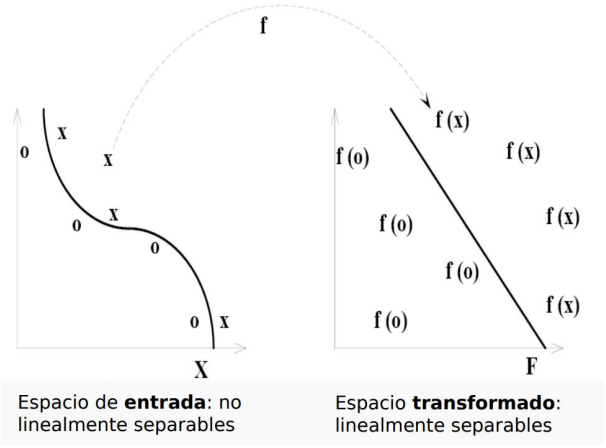

Figura 4.3: Transformación núcleo en SVM
Fuente: [31]

4.3. Implementación en juegos con un equilibrio de Nash

El primer paso de nuestro estudio, una vez elegido el tipo de juego que queríamos estudiar, fue comenzar por algo sencillo, aplicar los diferentes algoritmos en juegos que solo tuvieran un equilibrio de Nash posible. Para ello hicimos una modificación en nuestra función inicial para detectar los equilibrios de Nash de tal forma que filtrara las diferentes matrices creadas de forma aleatoria, seleccionando únicamente las que eran objetivo de nuestro estudio.

Probamos diferentes tamaños de conjuntos de datos iniciales para comprobar si el aumento del tamaño afectaba al porcentaje de acierto a la hora de predecir los equilibrios de Nash.

Una vez obtenido nuestro conjunto de datos filtrado, lo dividimos en un conjunto de entrenamiento, que serviría de aprendizaje para nuestros algoritmos, y un conjunto de prueba o test donde comprobamos el funcionamiento de dichos algoritmos.

Los resultados obtenidos están recogidos en la Tabla 4.1 y en la Figura 4.4.

Podemos observar que el porcentaje de aciertos se encuentra entre el 40 % y el 90 %. Los mejores resultados obtenidos fueron por parte de SVM y los peores por parte del perceptrón simple.

Además podemos notar cómo, en general, a partir de un tamaño de datos de 10.000 no parece mejorar en gran medida el porcentaje de acierto, pues empieza a crecer muy lentamente el gráfico. Podemos percibir una mejora más notable en SVM que parece seguir aumentando hasta un tamaño de alrededor de 50.000.

Otro de los aspectos que quisimos tener en cuenta fue el tiempo que invertía el algoritmo en realizar la predicción. Los resultados están recogidos en la Tabla 4.2 y en la Figura 4.5.

Algoritmo / Tamaño	Perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	42.8571 %	62.2857 %	69.1429 %	70.8571 %	57.1429 %	62.2857 %
Datos_10.000	54.3717 %	72.7273 %	72.2061 %	79.2125 %	63.3468 %	76.8384 %
Datos_25.000	55.9982 %	73.1752 %	73.9120 %	85.3788 %	68.8925 %	79.4842 %
Datos_50.000	55.2225 %	73.7215 %	73.8245 %	88.7427 %	71.7538 %	79.6934 %
Datos_70.000	53.9132 %	74.1039 %	74.2683 %	90.0197 %	72.2624 %	80.3025 %
Datos_80.000	57.9550 %	74.0584 %	73.9868 %	89.9828 %	72.6407 %	79.7293 %
Datos_90.000	59.6060 %	73.4907 %	73.4459 %	90.1509 %	73.0238 %	79.1123 %
Datos_100.000	58.3801 %	73.5699 %	73.5928 %	90.5537 %	73.2546 %	79.6744 %

Tabla 4.1: Resultados porcentaje de aciertos inicial

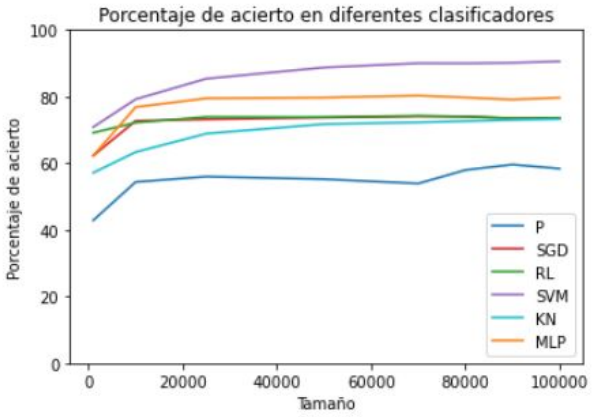

Figura 4.4: Gráfico porcentaje de aciertos inicial

Podemos ver cómo, al aumentar el tamaño, el tiempo necesario para obtener los resultados aumenta muy rápidamente, creciendo mucho más rápido en el SGD en comparación del resto de clasificadores. Un detalle a destacar es la evolución del perceptrón multicapa, pues una vez superados los 80.000 datos comienza a descender el tiempo necesario.

Los algoritmos más rápidos fueron el perceptrón, k-vecinos más próximos y regresión logística.

El algoritmo SVM, que antes habíamos observado que nos ofrecía el mejor resultado con respecto al porcentaje de aciertos, se encuentra en el punto medio entre los clasificadores más rápidos y más lentos en obtener el resultado, sin necesitar un tiempo excesivo.

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.029007 s	0.028005 s	0.129028 s	0.068018 s	0.007999 s	0.386090 s
Datos_10.000	0.118027 s	0.299068 s	0.222050 s	2.138490 s	0.201046 s	34.712901 s
Datos_25.000	0.202503 s	7.776922 s	0.472106 s	10.782435 s	0.990225 s	50.454172 s
Datos_50.000	0.409094 s	118.523738 s	0.964169 s	40.290098 s	4.145453 s	74.625539 s
Datos_70.000	1.130997 s	236.367094 s	1.301294 s	92.561718 s	7.218581 s	132.955636 s
Datos_80.000	0.815185 s	283.601767 s	1.709385 s	120.530445 s	9.315482 s	279.879577 s
Datos_90.000	0.796180 s	315.313101 s	1.856419 s	179.072576 s	12.603247 s	265.586643 s
Datos_100.000	1.132953 s	384.071434 s	2.043890 s	199.305745 s	16.334779 s	161.375684 s

Tabla 4.2: Resultados tiempos inicial

Tras ver estos resultados decidimos cambiar algunos parámetros de las funciones de los diferentes clasificadores para ver si mejoraban o empeoraban los resultados ya obtenidos. Estos cambios fueron: probar diferentes funciones núcleo en el clasificador SVM, variar el número de vecinos más próximos para la clasificación y variar el número de capas ocultas del perceptrón multicapa, pues al aumentarlo, es capaz de resolver problemas más complejos, también variamos el método que utilizaba para la actualización de los pesos.

En primer lugar mostraremos los resultados tras variar las funciones núcleo en el clasificador SVM recogidos en la Tabla 4.3 y en la Tabla 4.4.

Núcleo / Tamaño	Polinómico	RBF	Sigmoide
Datos_1.000	56.25 %	64.2045 %	67.0455 %
Datos_10.000	78.9294 %	82.2323 %	78.8155 %
Datos_100.000	88.636 %	90.3561 %	78.3212 %

Tabla 4.3: Aciertos variando los núcleos

Núcleo / Tamaño	Polinómico	RBF	Sigmoide
Datos_1.000	0.044 s	0.054 s	0.031 s
Datos_10.000	1.73 s	2.0115 s	1.1951 s
Datos_100.000	171.9046 s	182.491 s	143.2527 s

Tabla 4.4: Tiempos variando los núcleos

Podemos observar cómo los resultados no parecen variar demasiado al cambiar el tipo de

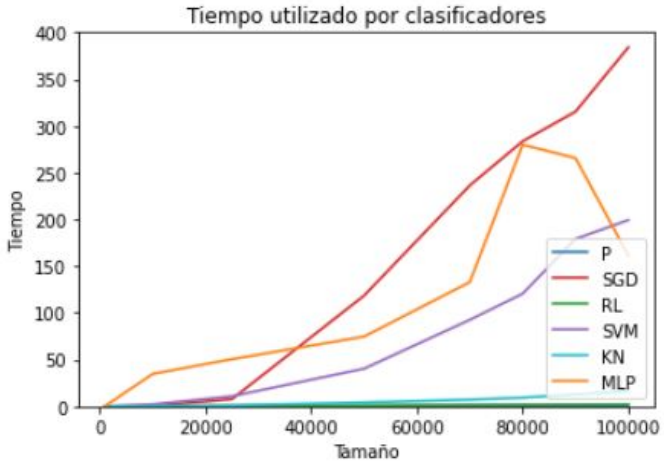

Figura 4.5: Resultados tiempo inicial

núcleo.

Respecto a variar el número de vecinos utilizado para la clasificación, los resultados fueron recogidos en la Tabla 4.5 y en la Tabla 4.6.

Nº Vecinos / Tamaño	1	3	5	7	9	11	13	15	17	19
Datos_1.000	40.9091 %	42.6136 %	44.8864 %	47.7273 %	52.8409 %	51.1364 %	54.5455 %	55.6818 %	56.25 %	52.8409 %
Datos_10.000	50.1139 %	52.9043 %	58.9977 %	62.9841 %	64.5786 %	66.7995 %	67.426 %	68.1663 %	69.0205 %	68.6788 %
Datos_100.000	57.8178 %	62.2097 %	67.8745 %	70.3228 %	72.4442 %	73.7974 %	74.6976 %	75.6608 %	76.3431 %	76.8534 %

Tabla 4.5: Porcentaje de aciertos en función del k

Tras observar estos resultados vemos cómo el porcentaje de acierto aumenta más significativamente hasta el vecino número 9 - 11 y posteriormente ya el porcentaje de acierto crece más lentamente.

Respecto al número de capas ocultas y al algoritmo utilizado para actualizar los pesos obtuvimos los resultados:

- Algoritmo lbfgs, resultados recogidos en la Tabla 4.7 y en la Tabla 4.8.

- Algoritmo sgd, resultados recogidos en la Tabla 4.9 y en la Tabla 4.10

Nº Vecinos / Tamaño	1	3	5	7	9	11	13	15	17	19
Datos_1.000	0.009 s	0.008 s	0.008 s	0.008 s	0.009 s	0.01 s	0.009 s	0.01 s	0.008 s	0.007 s
Datos_10.000	0.142 s	0.163 s	0.2 s	0.2111 s	0.214 s	0.2201 s	0.204 s	0.215 s	0.2271 s	0.219 s
Datos_100.000	10.5973 s	11.6769 s	15.5025 s	15.3632 s	14.3928 s	14.1732 s	14.5342 s	14.9223 s	14.981 s	14.515 s

Tabla 4.6: Tiempos en función del k

Nº Capas ocultas / Tamaño	5	10	15
Datos_1.000	36.9318 %	52.2727 %	50.0 %
Datos_10.000	50.9112 %	78.3599 %	92.6538 %
Datos_100.000	51.6541 %	80.2076 %	94.7537 %

Tabla 4.7: Porcentaje de aciertos algoritmo lbfgs

Observamos que el porcentaje de acierto mejora según añadimos capas, sin embargo, aumenta en gran medida el tiempo invertido hasta obtener el resultado, por lo que 10 capas parece ser la mejor opción. No parece haber grandes diferencias entre los distintos algoritmos empleados para el cálculo.

En este punto tenemos otros dos posibles caminos a seguir: ver qué tal se comportan los diferentes algoritmos en juegos 3x3 con un número arbitrario de equilibrios de Nash, es decir, dejando que nuestro conjunto de datos tenga de 0 a 9 equilibrios de Nash, y por otro lado, ver cómo se comportan los clasificadores para un único equilibrio de Nash pero en juegos con más estrategias por jugador.

4.3.1. Comportamiento en juegos con un único equilibrio de Nash pero con más de tres estrategias por jugador

En este procedimiento aumentamos el número de estrategias, lo cual implica un aumento en el tamaño de nuestra matriz de pagos del juego y con ello la cantidad de posibles posiciones en las que encontrar el equilibrio de Nash.

La cantidad de variables de nuestro problema aumenta de forma muy rápida, partiendo de las 18 variables iniciales de un juego 3x3, a contar, por ejemplo, en un juego 5x5, con 50 variables.

Esto nos hizo suponer que el tiempo necesario para solucionar estos problemas más complejos

Nº Capas ocultas / Tamaño	5	10	15
Datos_1.000	2.0585 s	1.4993 s	0.134 s
Datos_10.000	19.1122 s	21.7317 s	40.5775 s
Datos_100.000	216.9218 s	424.642 s	1087.8093 s

Tabla 4.8: Tiempo algoritmo lbfgs

Nº Capas ocultas / Tamaño	5	10	15
Datos_1.000	35.7955 %	54.5455 %	56.25 %
Datos_10.000	52.2779 %	79.385 %	88.5535 %
Datos_100.000	51.3503 %	79.8922 %	93.9224 %

Tabla 4.9: Porcentaje de aciertos algoritmo sgd

aumentaría y, a su vez, disminuiría el porcentaje de acierto del clasificador a la hora de predecir el equilibrio de Nash.

Tras este planteamiento inicial, comenzamos a utilizar los diferentes algoritmos de clasificación para ver si, con los resultados obtenidos, confirmábamos o no este hecho.

Juegos de 4 estrategias por jugador

Los resultados obtenidos para juegos de 4 estrategias respecto al porcentaje de aciertos están recogidos en la Tabla 4.11 y los podemos visualizar en la Figura 4.6.

En cuanto al tiempo empleado los resultados se encuentran en la Tabla 4.12, representados de forma gráfica en la Figura 4.7.

Juegos de 5 estrategias por jugador

Los resultados obtenidos para juegos de 5 estrategias respecto al porcentaje de aciertos podemos observarlos en la Tabla 4.13 y en la Figura 4.8.

En cuanto al tiempo empleado los resultados están recogidos en la Tabla 4.14 y la Figura 4.9.

Tamaño \ Nº Capas ocultas	5	10	15
Datos_1.000	3.3218 s	3.5858 s	4.094 s
Datos_10.000	13.713 s	10.7507 s	19.6905 s
Datos_100.000	30.7984 s	75.4501 s	100.977 s

Tabla 4.10: Tiempo algoritmo sgd

Tamaño \ Algoritmo	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	17.1233 %	41.7808 %	52.0548 %	47.2603 %	24.6575 %	26.7123 %
Datos_10.000	30.8609 %	64.7020 %	65.2318 %	65.7616 %	37.4172 %	45.0993 %
Datos_25.000	39.5183 %	67.4166 %	67.5754 %	71.3340 %	42.5093 %	48.6765 %
Datos_50.000	39.8741 %	68.1139 %	68.1532 %	73.9769 %	45.7896 %	50.0787 %
Datos_70.000	47.9442 %	67.6126 %	67.7718 %	75.5175 %	46.8858 %	50.2857 %
Datos_80.000	45.3109 %	68.1194 %	68.3675 %	76.1082 %	47.6265 %	49.5203 %
Datos_90.000	43.3907 %	67.8144 %	67.9603 %	76.1234 %	48.1690 %	49.4602 %
Datos_100.000	44.0682 %	68.0052 %	67.9396 %	76.4239 %	47.0801 %	49.4094 %

Tabla 4.11: Porcentaje de aciertos juegos de 4 estrategias por jugador

Juegos de 6 estrategias por jugador

Los resultados obtenidos para juegos de 6 estrategias respecto al porcentaje de aciertos se encuentran en la Tabla 4.15 y se pueden observar en la figura 4.10.

Los resultados respecto al tiempo se encuentran en la Tabla 4.16 y representados en la Figura 4.11.

Juegos de 7 estrategias por jugador

Los resultados para juegos de 7 estrategias sobre porcentaje de aciertos están recogidos en la Tabla 4.17 y representados en la Figura 4.12.

Los resultados sobre el tiempo necesario para la clasificación se encuentran en la Tabla 4.18 y

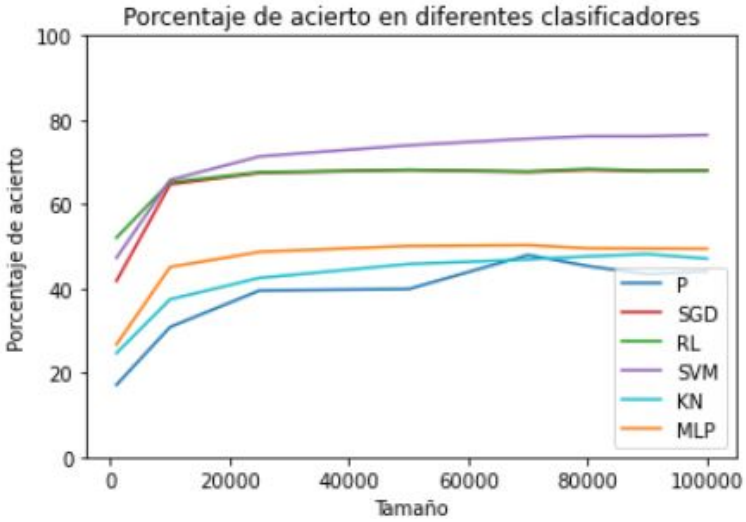

Figura 4.6: Gráfico porcentaje de aciertos juegos de 4 estrategias por jugador

en la Figura 4.13.

Juegos de 8 estrategias por jugador

Los resultados obtenidos para juegos de 8 estrategias respecto al porcentaje de aciertos se encuentran en la Tabla 4.19 y representados de forma gráfica en la Figura 4.14.

En cuanto al tiempo empleado los resultados están recogidos en la Tabla 4.20 y en la Figura 4.15.

Juegos de 9 estrategias por jugador

Los resultados para juegos de 9 estrategias sobre el porcentaje de aciertos están recogidos en la Tabla 4.21 y en la Figura 4.16.

En cuanto al tiempo empleado los resultados se encuentran en la Tabla 4.22 y representados en la Figura 4.17.

Juegos de 10 estrategias por jugador

Los resultados obtenidos para juegos de 10 estrategias respecto al porcentaje de aciertos se encuentran en la Tabla 4.23 y en la Figura 4.18.

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.047008 s	0.059012 s	1.158607 s	0.071014 s	0.007003 s	1.729385 s
Datos_10.000	0.253058 s	0.511116 s	0.520112 s	2.660594 s	0.152034 s	49.650606 s
Datos_25.000	0.881195 s	2.506561 s	0.756168 s	13.232116 s	0.789179 s	152.274431 s
Datos_50.000	1.781393 s	10.236288 s	1.399313 s	50.248668 s	3.225719 s	299.432702 s
Datos_70.000	2.114469 s	31.860267 s	2.015234 s	117.971585 s	5.808983 s	448.522128 s
Datos_80.000	2.133467 s	36.189149 s	2.226499 s	138.830997 s	7.469663 s	723.106831 s
Datos_90.000	2.471053 s	47.824802 s	1.909430 s	142.239099 s	7.913763 s	488.720844 s
Datos_100.000	2.376510 s	67.819191 s	2.049237 s	191.787358 s	9.997955 s	444.892097 s

Tabla 4.12: Tiempo juegos de 4 estrategias por jugador

En cuanto al tiempo empleado los resultados se encuentran en la Tabla 4.24 y en el gráfico 4.19.

A lo largo de todas las tablas hemos podido comprobar cómo, por lo general, en los diferentes clasificadores el tiempo iba aumentando a medida que aumentaba el número de estrategias del juego y cómo el porcentaje disminuía también notablemente. Esto hace que podamos confirmar nuestra hipótesis inicial.

4.3.2. Juegos de tres estrategias por jugador, sin distinguir el número de equilibrios de Nash

Los resultados obtenidos para juegos de 3 estrategias sin filtrar en función del número de equilibrios que tuvieran en el primer fichero generado aleatoriamente, respecto al porcentaje de aciertos los tenemos recogidos en la Tabla 4.25 y representados en la Figura 4.20.

En cuanto al tiempo empleado, los resultados se encuentran en la Tabla 4.26 y en la Figura 4.21.

Podemos notar cómo los resultados también son “peores”en comparación con los juegos en los que previamente habíamos filtrado el número de equilibrios de Nash, pues el tiempo aumenta y el porcentaje de acierto disminuye.

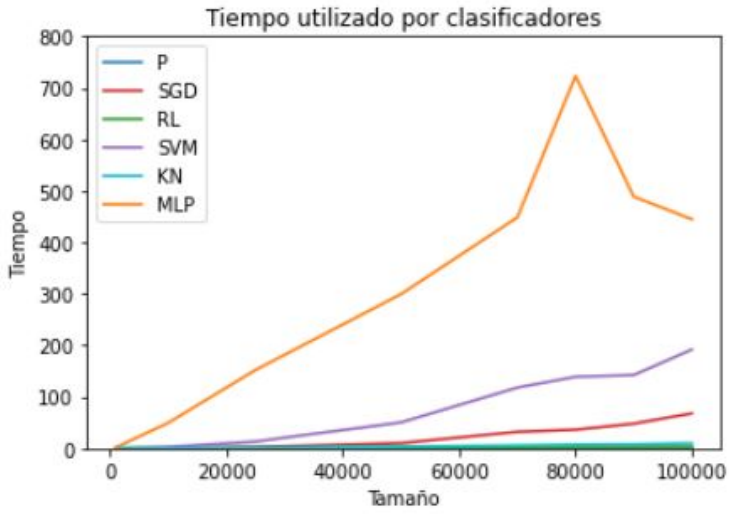

Figura 4.7: Gráfico de tiempos de juegos de 4 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	9.1549 %	21.8310 %	21.8310 %	22.5352 %	11.9718 %	9.8592 %
Datos_10.000	20.3795 %	58.1167 %	60.1546 %	54.1110 %	23.2607 %	23.0499 %
Datos_25.000	24.9858 %	60.3419 %	60.6268 %	60.0570 %	23.8462 %	31.1966 %
Datos_50.000	29.6787 %	61.4008 %	61.7813 %	61.7954 %	26.4233 %	32.4690 %
Datos_70.000	32.2998 %	62.2765 %	62.4684 %	63.6097 %	27.3205 %	33.2997 %
Datos_80.000	34.3231 %	63.5523 %	63.2767 %	64.8235 %	27.9047 %	33.2296 %
Datos_90.000	32.7984 %	63.0320 %	63.0556 %	65.3292 %	28.3849 %	32.4443 %
Datos_100.000	34.1612 %	63.5751 %	63.4473 %	65.5478 %	28.1791 %	32.3872 %

Tabla 4.13: Porcentaje de aciertos juegos de 5 estrategias por jugador

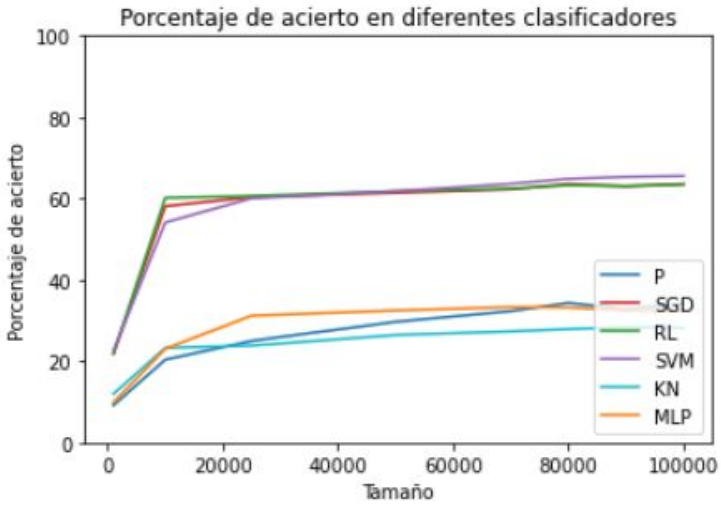

Figura 4.8: Gráfico porcentaje de aciertos juegos de 5 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.084019 s	0.103021 s	0.850190 s	0.063014 s	0.006002 s	0.606134 s
Datos_10.000	0.386724 s	0.805894 s	0.818834 s	3.220229 s	0.116006 s	47.216137 s
Datos_25.000	1.256794 s	3.358323 s	1.545344 s	14.836279 s	0.623644 s	108.861647 s
Datos_50.000	2.772099 s	10.633528 s	1.727880 s	49.894179 s	2.276234 s	328.721621 s
Datos_70.000	4.024230 s	19.541764 s	3.103692 s	96.859522 s	4.467813 s	436.945232 s
Datos_80.000	4.444854 s	24.023358 s	3.066686 s	127.185251 s	5.741814 s	474.075426 s
Datos_90.000	5.350591 s	32.345808 s	3.863510 s	194.388919 s	7.431391 s	721.040985 s
Datos_100.000	5.588368 s	33.756331 s	3.723831 s	233.784718 s	8.859026 s	642.139466 s

Tabla 4.14: Tiempo juegos de 5 estrategias por jugador

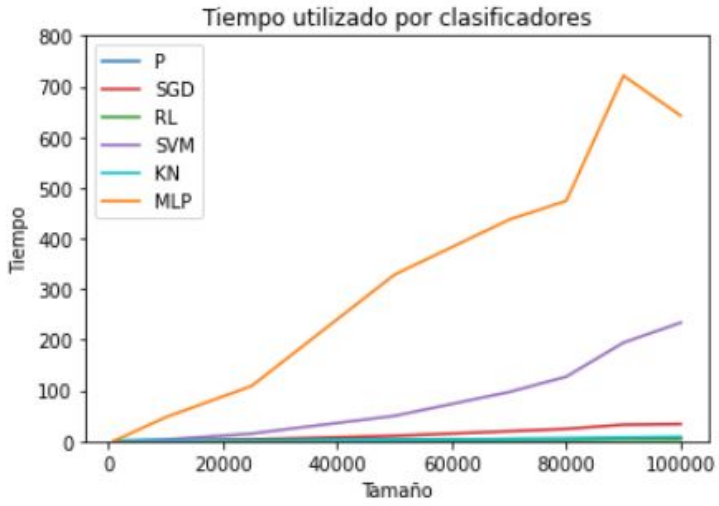

Figura 4.9: Gráfico tiempo juegos de 5 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	5.1095 %	9.4891 %	22.6277 %	8.7591 %	10.2190 %	7.2993 %
Datos_10.000	12.0920 %	47.0326 %	46.2166 %	38.7240 %	13.6499 %	11.7211 %
Datos_25.000	18.6723 %	54.8348 %	55.5017 %	49.5908 %	14.4589 %	18.4298 %
Datos_50.000	21.4745 %	58.8007 %	58.8156 %	55.3462 %	15.6273 %	22.0278 %
Datos_70.000	22.0871 %	58.9309 %	58.8242 %	56.3380 %	17.0294 %	23.5169 %
Datos_80.000	23.6247 %	58.4846 %	58.8011 %	56.3902 %	16.7086 %	22.8521 %
Datos_90.000	24.0194 %	59.3619 %	58.8110 %	56.9197 %	17.2519 %	22.5146 %
Datos_100.000	24.9815 %	59.1604 %	58.9675 %	57.8475 %	17.4010 %	22.3780 %

Tabla 4.15: Porcentaje de aciertos juegos de 6 estrategias por jugador

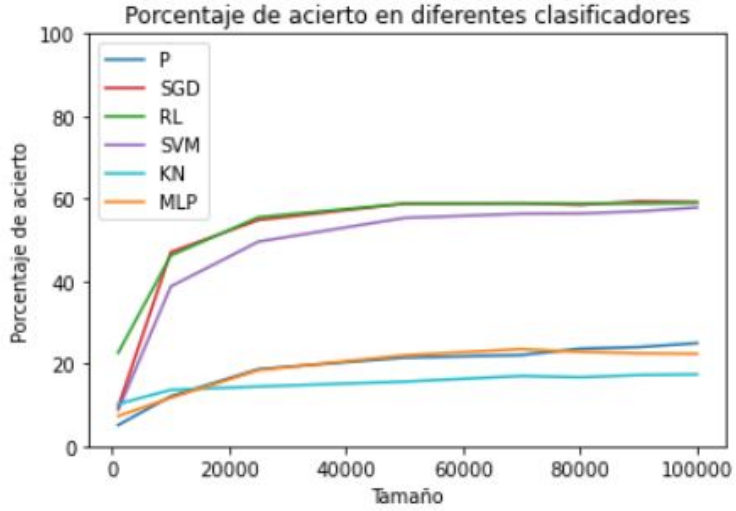

Figura 4.10: Gráfico porcentaje de aciertos juegos de 6 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.129029 s	0.156703 s	0.606830 s	0.073020 s	0.007007 s	1.353292 s
Datos_10.000	0.935209 s	1.891421 s	9.439038 s	3.847641 s	0.114016 s	31.974137 s
Datos_25.000	2.658591 s	7.657129 s	4.151927 s	18.992961 s	0.570140 s	96.609731 s
Datos_50.000	5.404301 s	20.086082 s	3.431771 s	58.550270 s	2.124561 s	407.243980 s
Datos_70.000	7.335960 s	29.343357 s	3.600847 s	108.332239 s	3.762784 s	466.854893 s
Datos_80.000	8.632490 s	36.490095 s	3.605167 s	143.232543 s	4.500968 s	581.222238 s
Datos_90.000	9.396193 s	43.598394 s	4.136925 s	188.580209 s	6.437669 s	802.246320 s
Datos_100.000	10.563003 s	53.100879 s	5.150152 s	246.613117 s	7.680027 s	785.520276 s

Tabla 4.16: Tiempo juegos de 6 estrategias por jugador

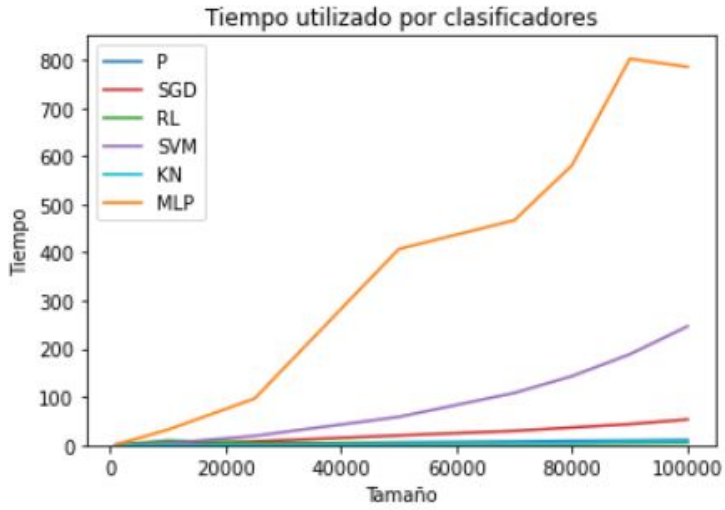

Figura 4.11: Gráfico tiempo juegos de 6 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	3.9683 %	8.7302 %	13.4921 %	5.5556 %	7.1429 %	3.9683 %
Datos_10.000	9.3726 %	36.7157 %	33.6174 %	29.9768 %	8.2107 %	6.8164 %
Datos_25.000	11.6379 %	48.7069 %	48.4606 %	41.2869 %	9.9138 %	10.1293 %
Datos_50.000	14.9454 %	52.9013 %	52.9167 %	46.5753 %	9.6968 %	13.3754 %
Datos_70.000	15.5938 %	55.1267 %	54.9731 %	49.5888 %	10.1656 %	15.8899 %
Datos_80.000	16.8996 %	56.0108 %	55.8566 %	50.6893 %	10.9804 %	15.5114 %
Datos_90.000	16.3516 %	55.3617 %	55.5680 %	50.7304 %	10.8266 %	13.6192 %
Datos_100.000	17.8798 %	55.7061 %	55.7444 %	51.2438 %	10.9376 %	16.3414 %

Tabla 4.17: Porcentaje de aciertos juegos de 7 estrategias por jugador

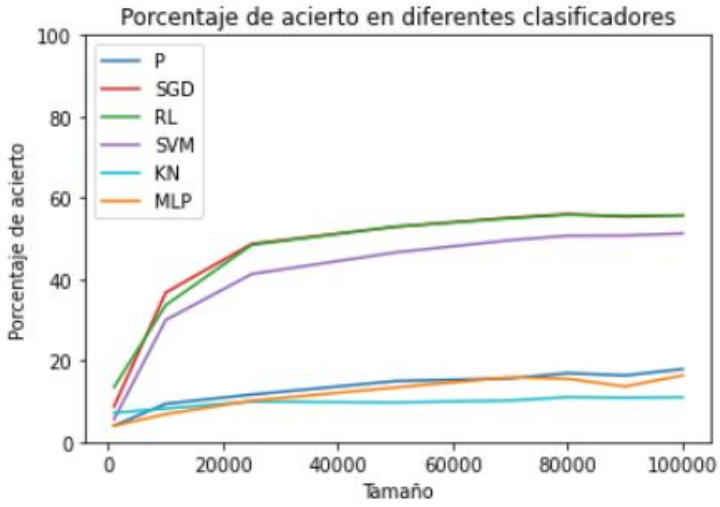

Figura 4.12: Gráfico porcentaje de aciertos juegos de 7 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.197044 s	0.243055 s	0.254056 s	0.062014 s	0.006002 s	4.152759 s
Datos_10.000	1.447298 s	2.360529 s	37.343342 s	3.940902 s	0.098001 s	123.509339 s
Datos_25.000	5.657099 s	12.470613 s	13.540091 s	27.074703 s	0.579131 s	224.187184 s
Datos_50.000	11.770393 s	40.510682 s	10.683149 s	85.784654 s	2.339523 s	454.132424 s
Datos_70.000	16.131142 s	68.853437 s	6.928010 s	155.093160 s	4.528013 s	546.932797 s
Datos_80.000	17.098026 s	73.385584 s	10.406405 s	200.049118 s	5.626350 s	1048.361808 s
Datos_90.000	18.586171 s	82.921719 s	9.380097 s	272.631285 s	7.075464 s	913.873519 s
Datos_100.000	21.972948 s	107.502796 s	9.934221 s	370.617215 s	9.062170 s	1284.244221 s

Tabla 4.18: Tiempo juegos de 7 estrategias por jugador

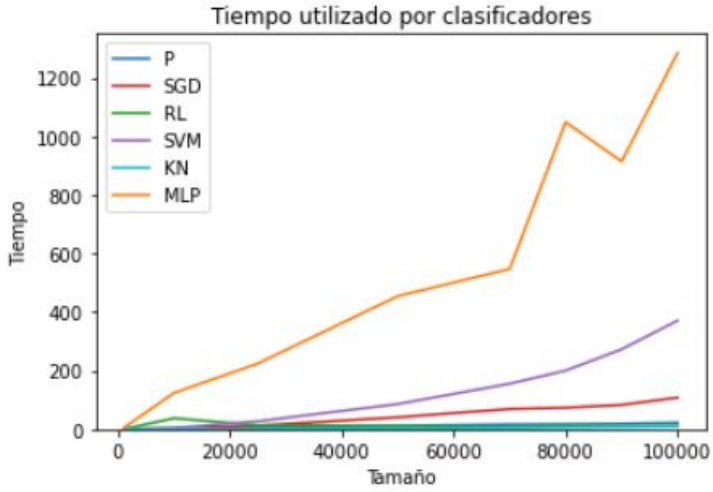

Figura 4.13: Gráfico tiempo juegos de 7 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	1.6129 %	1.6129 %	6.4516 %	4.0323 %	4.0323 %	0.0000 %
Datos_10.000	6.6929 %	25.3543 %	24.2520 %	21.4961 %	5.9843 %	4.4882 %
Datos_25.000	7.1834 %	41.1468 %	36.1059 %	32.9868 %	5.5766 %	5.1355 %
Datos_50.000	10.7048 %	47.2105 %	46.0228 %	38.2872 %	7.3136 %	7.4387 %
Datos_70.000	10.1655 %	50.8613 %	49.8753 %	42.6224 %	6.9243 %	7.3663 %
Datos_80.000	11.1973 %	50.5986 %	50.8734 %	43.6506 %	6.5849 %	9.1658 %
Datos_90.000	11.4601 %	50.6080 %	50.2318 %	43.5307 %	6.8673 %	9.7279 %
Datos_100.000	11.9307 %	52.2059 %	51.9924 %	44.7660 %	6.7916 %	9.8751 %

Tabla 4.19: Porcentaje de aciertos juegos de 8 estrategias por jugador

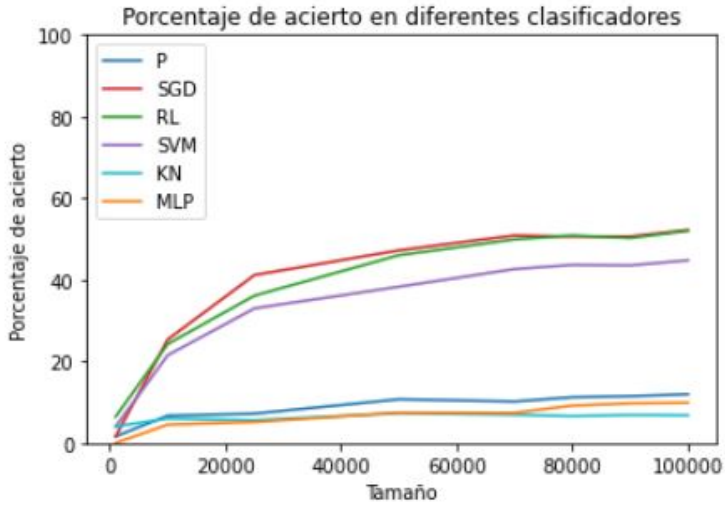

Figura 4.14: Gráfico porcentaje de aciertos juegos de 8 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.304068 s	0.438942 s	0.597033 s	0.107916 s	0.008002 s	3.304544 s
Datos_10.000	3.515938 s	5.245440 s	11.481567 s	4.669136 s	0.112026 s	121.785480 s
Datos_25.000	10.055041 s	23.691473 s	186.036696 s	29.846259 s	0.649667 s	538.685843 s
Datos_50.000	19.647643 s	63.175964 s	17.895118 s	117.952891 s	2.251505 s	500.853604 s
Datos_70.000	25.924621 s	106.020345 s	29.904632 s	203.596181 s	3.964885 s	726.331763 s
Datos_80.000	34.748883 s	129.122730 s	34.107979 s	261.126391 s	5.495229 s	699.883684 s
Datos_90.000	33.658443 s	138.819313 s	36.102659 s	332.252793 s	6.971119 s	717.328750 s
Datos_100.000	39.282013 s	159.514496 s	15.774174 s	405.208936 s	7.725306 s	837.828569 s

Tabla 4.20: Tiempo juegos de 8 estrategias por jugador

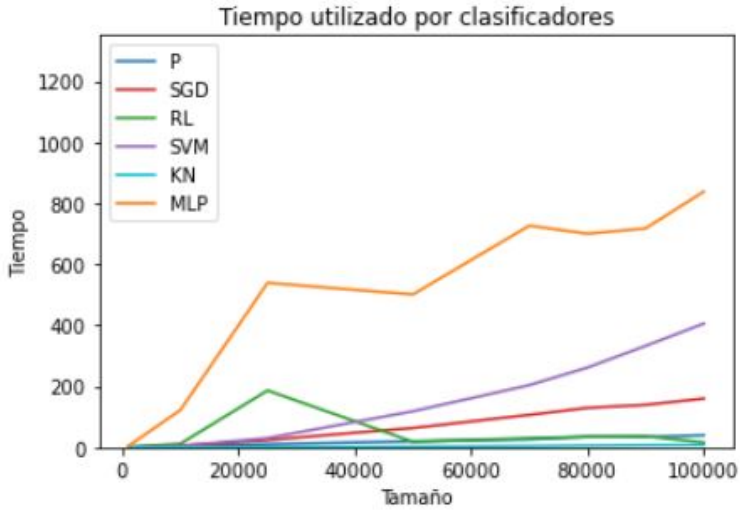

Figura 4.15: Gráfico tiempo juegos de 8 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	1.6260 %	1.6260 %	3.2520 %	0.8130 %	2.4390 %	1.6260 %
Datos_10.000	3.3515 %	17.7708 %	16.7576 %	12.6267 %	4.1309 %	2.3383 %
Datos_25.000	5.3359 %	30.5047 %	29.2832 %	23.6258 %	4.4359 %	4.1144 %
Datos_50.000	6.5526 %	41.2363 %	40.3970 %	31.1168 %	4.3415 %	4.3577 %
Datos_70.000	7.8105 %	45.3034 %	45.0074 %	35.1361 %	5.0097 %	4.9755 %
Datos_80.000	8.3350 %	47.3116 %	47.0018 %	38.2770 %	4.7971 %	5.2868 %
Datos_90.000	7.8720 %	47.3032 %	47.1427 %	37.3808 %	4.9835 %	5.1707 %
Datos_100.000	9.3559 %	48.1770 %	47.9602 %	38.4115 %	4.5535 %	5.1879 %

Tabla 4.21: Porcentaje de aciertos juegos de 9 estrategias por jugador

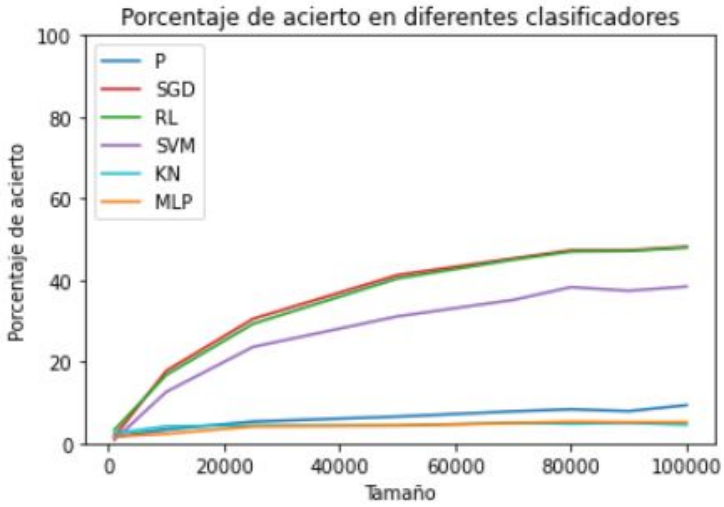

Figura 4.16: Gráfico porcentaje de aciertos juegos de 9 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.393096 s	0.691155 s	0.935207 s	0.122030 s	0.008999 s	15.205510 s
Datos_10.000	7.316626 s	7.578596 s	12.459795 s	5.296414 s	0.125027 s	120.542374 s
Datos_25.000	16.702344 s	38.369061 s	96.384587 s	32.841752 s	0.555124 s	700.848462 s
Datos_50.000	33.595858 s	98.431324 s	56.148623 s	129.258929 s	2.142479 s	1070.079973 s
Datos_70.000	43.144017 s	152.599326 s	51.618593 s	243.346534 s	4.063771 s	553.071787 s
Datos_80.000	34.654754 s	130.168123 s	29.241976 s	281.337803 s	4.606611 s	394.782543 s
Datos_90.000	38.934569 s	157.184892 s	41.163874 s	354.701779 s	5.664259 s	619.686995 s
Datos_100.000	43.977271 s	170.492777 s	21.262221 s	439.988606 s	6.185448 s	567.398287 s

Tabla 4.22: Tiempo juegos de 9 estrategias por jugador

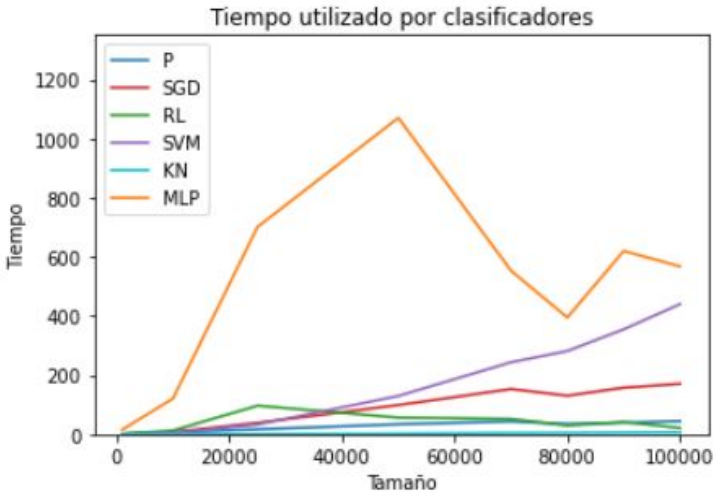

Figura 4.17: Gráfico tiempo juegos de 9 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	1.6667 %	2.5000 %	2.5000 %	0.8333 %	1.6667 %	0.8333 %
Datos_10.000	2.4741 %	11.6520 %	11.1732 %	7.7414 %	2.2346 %	2.5539 %
Datos_25.000	3.0699 %	22.5996 %	21.1953 %	16.7864 %	2.9066 %	2.6453 %
Datos_50.000	4.7974 %	34.0519 %	30.8428 %	25.8023 %	2.9335 %	3.1280 %
Datos_70.000	5.5446 %	39.1486 %	38.5106 %	28.9178 %	3.2711 %	3.1319 %
Datos_80.000	5.8973 %	41.2100 %	40.5186 %	30.7982 %	3.5079 %	3.6807 %
Datos_90.000	5.8575 %	42.8019 %	42.4600 %	31.1949 %	3.2931 %	3.4821 %
Datos_100.000	6.5039 %	43.5284 %	43.0690 %	32.7289 %	3.5864 %	3.6186 %

Tabla 4.23: Porcentaje de aciertos juegos de 10 estrategias por jugador

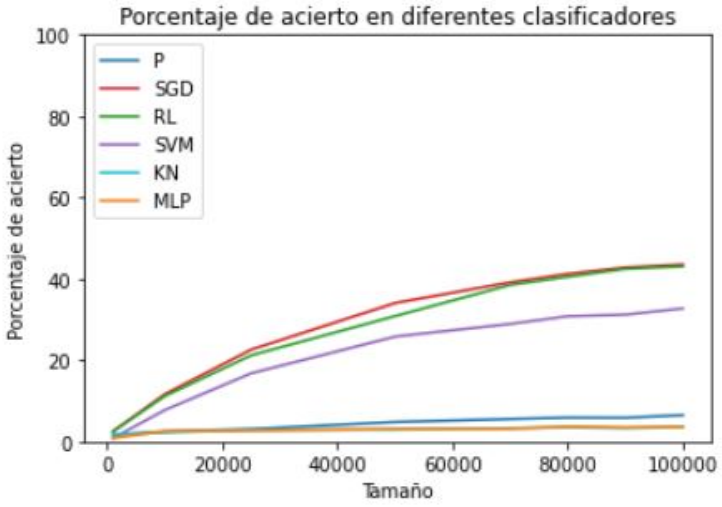

Figura 4.18: Gráfico porcentaje de aciertos juegos de 10 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.422796 s	0.655635 s	0.840194 s	0.106015 s	0.007002 s	14.168771 s
Datos_10.000	8.955952 s	6.535888 s	13.640850 s	5.157512 s	0.100318 s	155.810672 s
Datos_25.000	20.324278 s	31.163762 s	67.110016 s	30.579340 s	0.520596 s	289.498235 s
Datos_50.000	36.698158 s	97.584066 s	166.858863 s	126.894383 s	1.864899 s	1610.850622 s
Datos_70.000	67.819655 s	196.873574 s	171.409241 s	278.389359 s	3.934956 s	1490.291420 s
Datos_80.000	81.202907 s	265.807852 s	147.608008 s	350.532416 s	5.378403 s	957.033366 s
Datos_90.000	79.644150 s	277.161579 s	83.846646 s	452.912874 s	7.286386 s	1069.140797 s
Datos_100.000	91.340284 s	360.521471 s	109.148307 s	508.297373 s	6.663016 s	506.820540 s

Tabla 4.24: Tiempo juegos de 10 estrategias por jugador

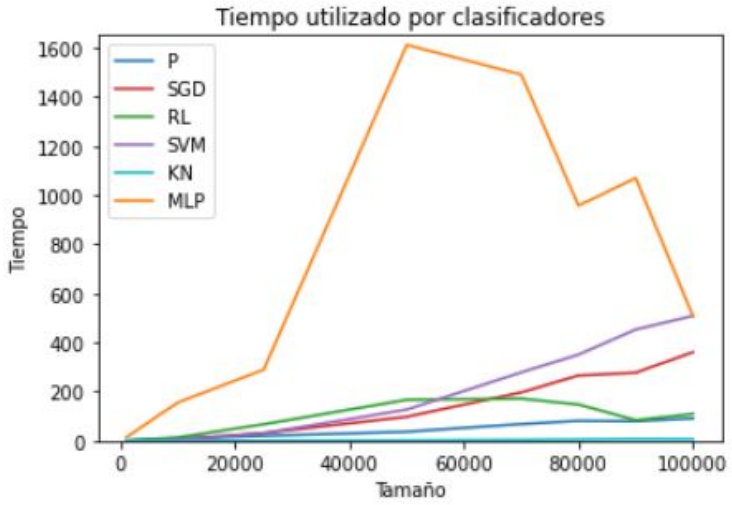

Figura 4.19: Gráfico tiempo juegos de 10 estrategias por jugador

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	16.0000 %	35.6667 %	38.3333 %	40.6667 %	28.6667 %	31.6667 %
Datos_10.000	22.0333 %	42.3000 %	50.8667 %	57.8000 %	40.3000 %	56.9333 %
Datos_25.000	20.5333 %	42.1467 %	50.5333 %	63.7333 %	44.6400 %	57.0000 %
Datos_50.000	24.9533 %	43.6067 %	51.8333 %	67.9467 %	47.1867 %	58.9000 %
Datos_70.000	25.0524 %	43.1476 %	51.1476 %	70.0524 %	47.2667 %	57.9429 %
Datos_80.000	27.2500 %	43.9667 %	51.8625 %	71.0125 %	48.7167 %	58.6417 %
Datos_90.000	27.0519 %	43.3259 %	51.5074 %	71.2926 %	48.7741 %	58.8741 %
Datos_100.000	27.0167 %	44.1567 %	51.7900 %	72.2567 %	49.4667 %	57.9633 %

Tabla 4.25: Porcentaje de aciertos sin distinguir el número de equilibrios de Nash

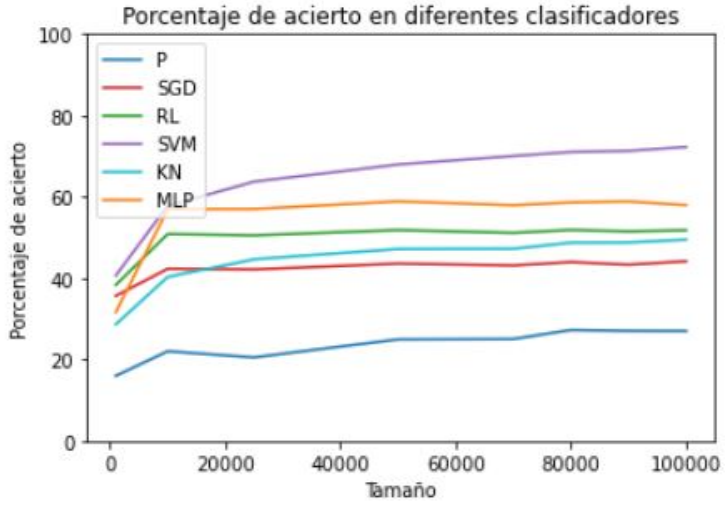

Figura 4.20: Gráfico porcentaje de aciertos sin distinguir el número de equilibrios de Nash

Algoritmo / Tamaño	perceptrón	SGD	R.Logística	SVM	K-Vecinos	P.Multicapa
Datos_1.000	0.089446 s	0.099021 s	10.067859 s	0.180056 s	0.011005 s	6.107368 s
Datos_10.000	0.410304 s	1.193934 s	61.272451 s	8.530184 s	0.435104 s	100.578990 s
Datos_25.000	1.046209 s	44.507413 s	89.440958 s	44.227618 s	2.583879 s	273.041815 s
Datos_50.000	3.618819 s	214.493454 s	217.338301 s	195.158879 s	9.870903 s	926.308680 s
Datos_70.000	5.485997 s	346.664026 s	182.073686 s	391.622077 s	19.303275 s	810.950815 s
Datos_80.000	5.827775 s	409.218096 s	264.287596 s	520.394186 s	24.708145 s	1143.222273 s
Datos_90.000	6.463646 s	469.917298 s	255.555806 s	665.320276 s	32.541057 s	916.229762 s
Datos_100.000	8.283345 s	579.143752 s	395.030312 s	843.505553 s	39.247890 s	1883.612772 s

Tabla 4.26: Tiempo sin distinguir el número de equilibrios de Nash

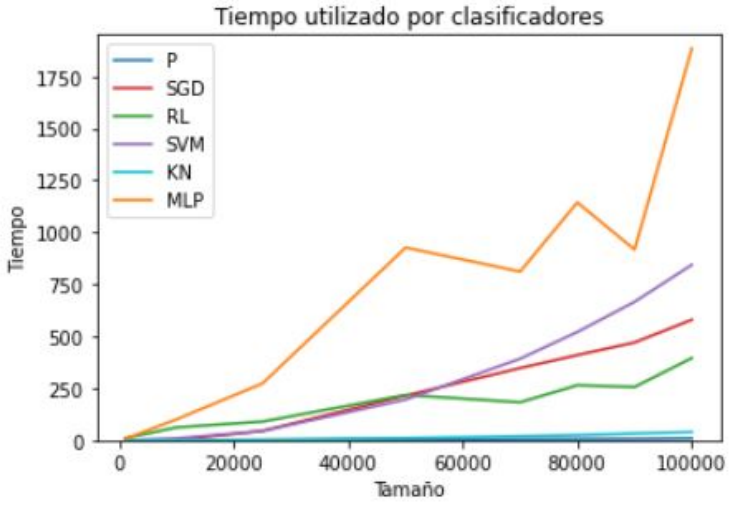

Figura 4.21: Gráfico tiempo sin distinguir el número de equilibrios de Nash

Capítulo 5

Conclusiones

Como hemos podido observar a través de las tablas en función del número de estrategias de nuestro juego, los tiempos, por lo general, iban aumentando y ofreciéndonos unos peores resultados respecto al porcentaje de acierto a la hora de predecir los equilibrios, si los comparamos con nuestras tablas iniciales en las cuales el juego constaba únicamente con 3 estrategias por jugador.

También podemos apreciar cómo al no filtrar nuestro fichero inicial de datos, para no distinguir el número de equilibrios de los que constan nuestros juegos ocurre algo similar, los tiempos aumentan considerablemente al aumentar el tamaño muestral y el porcentaje de acierto disminuye bastante.

Desde el punto de vista de los diferentes clasificadores cabe destacar el buen funcionamiento de SVM que nos ofrece siempre los mejores resultados de acuerdo con el porcentaje de aciertos. El segundo clasificador con mejor resultado es el algoritmo SGD y el tercero la regresión logística.

También es destacable el perceptrón multicapa en los juegos donde no filtramos el número de equilibrios de Nash, sin embargo, cuando aumentamos el número de estrategias no parece comportarse bien. Por otro lado, el perceptrón multicapa necesita mucho más tiempo que el resto de clasificadores para llegar a obtener la predicción, algo importante a considerar si queremos tener en cuenta la eficiencia.

El perceptrón simple y K-vecinos son los más rápidos en cuanto al tiempo necesario para la clasificación, pero por el contrario, su porcentaje de acierto es, por lo general, inferior al del resto de los clasificadores.

Como conclusión, teniendo en cuenta ambos criterios, tiempo y porcentaje de aciertos podríamos decir que el mejor clasificador es el SVM y que, en nuestro problema inicial, juegos de 3 estrategias con un único equilibrio de Nash todos los clasificadores nos han proporcionado unos

resultados aceptables.

Tenemos varias lineas de investigación abiertas para posibles trabajos futuros que podrían ser interesantes: intentar mejorar la eficiencia de los clasificadores, principalmente del perceptrón multicapa, comprobar el funcionamiento de los clasificadores variando sus parámetros en los problemas más complejos y no filtrar el número de equilibrios de Nash en este tipo de problemas donde las estrategias aumentan, entre otras.

Bibliografía

[1] CERDÁ TENA, E., PÉREZ NAVARRO, J. y JIMENO PASTOR, J. L., 2004, *Teoría de juegos*, PEARSON EDUCACIÓN, S.A.

[2] MONTMORT, P. R., 1713, *Essay d'analyse sur les jeux de hasard*, Imprimeur-Juré-Libraire de l'Univerfité, rue Galande

[3] ROCHESTER, N., HOLLAND, J. H., DUDA, W.L. y HABIT, L.H., 1956, *Tests on a cell assembly theory of the action of the brain, using a large digital computer*, IRE Transactions on Information Theory, Volumen: 2, Número: 3, 80-93

[4] OLDING HEBB, D., 1949, *The organization of behavior; a neuropsycholocigal theory*, A Wiley Book in Clinical Psychology, 62-78 .

[5] VON NEUMANN, J. y MORGENSTERN, O, 1944, *Theory of Games and Economic Behavior*, Princeton University Press

[6] WILHELM LEIBNIZ, G, 1704, *Nouveaux Essais sur l'entendement humain*

[7] http://dianainteligenciaartificial.blogspot.com/2015/07/red-adaline.html

[8] http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm

[9] http://www.cs.us.es/ fsancho/?e=77

[10] http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

[11] https://aws.amazon.com/es/what-is/neural-network/

[12] https://blog.damavis.com/perceptron-simple-definicion-matematica-y-propiedades/

[13] https://centrocompetencia.com/teoria-de-juegos/#: :text=B %C3 %A1sicamente %20es %20utilizada %20para %20modelar,puede %20esperar %20en %20cada %20situaci %C3 %B3n

[14] https://documat.unirioja.es

[15] https://es.mathworks.com/discovery/support-vector-machine.html

[16] https://gamco.es/glosario/adaline/

[17] https://interactivechaos.com/es/manual/tutorial-de-deep-learning/el-perceptron-multicapa

[18] https://leotronics.eu/es/nuestro-blog/inteligencia-artificial-y-redes-neuronales-artificiales

[19] https://policonomics.com/es

[20] https://rstudio-pubs-static.s3.amazonaws.com/897840_583523fb516541c6adfa4bcb8b573f89.html

[21] https://steemit.com/spanish/@iars.geo/breve-historias-de-las-redes-neuronales-artificiales-articulo-1

[22] https://unipython.com/curso-en-multicapa-perceptrones/?utm_content=cmp-true

[23] https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

[24] https://www.cienciadedatos.net/documentos/py17-regresion-logistica-python.html

[25] https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/

[26] https://www.elblogsalmon.com/conceptos-de-economia/que-es-la-teoria-de-juegos

[27] https://www.ibm.com/docs/es/spss-modeler

[28] https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model

[29] https://www.ibm.com/es-es/topics/knn#: :text=El %20algoritmo %20de %20k %20vecinos %20m %C3 %A1s %20cercanos %2C %20tambi %C3 %A9n %20conocido %20como,un %20punto %20de %20datos %20individual.

[30] https://www.ibm.com/es-es/topics/neural-networks

[31] https://www.infor.uva.es/ calonso/MUI-TIC/MineriaDatos/SVM.pdf

[32] https://www.unir.net/empresa/revista/teoriadejuegos/#: :text=La %20teor %C3 %ADa %20de %20juegos %20es,las %20elecciones %20de %20otros %20individuos

[33] https://www.unir.net/ingenieria/revista/redes-neuronales-artificiales/