UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

86.54 - Redes Neuronales
1er Cuatrimestre 2022

Trabajo Práctico 2 - Perceptrón

Alumnos:

Fernando Fraga - Legajo: 102369 - ffraga@fi.uba.ar

Profesores:

Juan Camilo Mininni

Ing. Sergio Eduardo Lew

Ing. Ricardo Veiga

Fecha de Entrega: 21/07/2022

Índice

1. Ejercicio 1

1.1. Enunciado

1. Implemente un perceptrón simple que aprenda la función lógica AND de 2 y de 4 entradas. Lo mismo para la función lógica OR. Para el caso de 2 dimensiones, grafique la recta discriminadora y todos los vectores de entrada de la red.

1.2. Perceptrón

Es una red neuronal con capas *feed-forward*, estas poseen una capa de entrada cuyo único rol es alimentar los patrones al resto de la red. Esta red puede tener no capas intermedias con unidades de procesamiento, seguído por una capa con las unidades de salida que representan el resultado de la computación.

Al ser restrictivamente *feed-forward*, existen conexiones únicamente con las unidades de la capa siguiente.

Las unidades de la capa intermedia se hacen llamar ünidades ocultasza que no posen una conexión directa con el exterior, ni con la entrada ni la salida.

A lo largo de este informe, la cantidad de capas será el numero de capas intermedias y la capa de salida, es decir, un perceptrón simple, de una capa, es una red que no posee capas intermedias y posee una única capa de conexiones (entre la entrada y la salida).

Por definición, las matrices de conexiones son asimétricas ya que son conexiones unidireccionales.

1.3. Perceptrón simple

Un perceptrón simple, es aquella red que posee una capa de conexiones feed-forward. Posee un set de N entradas y una capa de salida, sin poseer capas intermedias.

Llamando a una entrada k como ξ_k y la salida i O_i, la computación de una salida es descripta simplemente como

$$O_i = g(\sum_k w_{ik}\xi_k)$$

$$\mathbf{O} = g(\mathbf{W}\xi)$$

donde g() es la función de activación, dicha función suele ser no-lineal. La salida es una función explicita de la entrada, ya que la entrada es propagada a través de la red y produce la salida correspondiente.

Los umbrales θ pueden ser tomados como una conexión como una entrada que esta constantemente fijado con el valor $\xi_0 = -1$ y elegir los pesos sinápticos de tal manera que $w_{i0} = \theta_i$

Cabe destacar que vamos a resolver ejercicios de **hetero-asociación** donde las salidas de la red son distintas a los patrones de entrada. Estos incluyen problemas de *clasificación* de entradas en categorías.

1.4. Aprendizaje

Como fue mencionado anteriormente, es una red con aprendizaje **supervisado** en donde la salida de la red es compara con las respuestas correctas conocidas y recibe retroalimentación sobre los errores.

El aprendizaje comienza con un los pesos sinápticos de manera aleatoria, y luego van realizando mejoras sucesivas hasta alcanzar la solución en un número finito de pasos.

La idea del aprendizaje es tener unas entradas y salidas conocidas de entrenamiento, donde dichas entradas se aplican a la red y comparamos la salida de la red con la salida correcta, y luego cambiamos las fuerzas de conexión de tal manera que se minimice la diferencia, el error.

Este procedimiento suele hacerse de manera incremental, haciendo pequeños ajustes en respuesta de cada par de entrenamiento que se conoce con alta fidelidad.

Luego resulta interesante ingresas patrones que no son del conjunto de entrenamiento y observar si la red pudo generalizar lo aprendido.

1.5. Recta discriminadora

No todos los problemas pueden resolverse, especialmente si aplicamos unidades 'threshold' cuya función de activación cumple con:

$$g(\mathbf{h}) = sgn(\mathbf{h})$$

siendo $\mathbf{h} = \mathbf{Wxi}$, por lo que las salidas toman dos valores posibles {-1,1} y lo único que importa es que el signo de $\mathbf{h}$ sea igual a la salida deseada.

Al trabajar de forma matricial, podemos notar que se debe elegir cada vector de pesos correspondiente a una unidad de salida $\mathbf{w}$ de tal manera que el vector de entradas tenga su proyección a $\mathbf{w}$ el mismo signo que la salida deseada.

El plano que separa las proyecciones, hacia $\mathbf{w}$, positivas de las negativas es el plano $\mathbf{w}\xi = 0$, que pasa por el origen y es perpendicular a $\mathbf{w}$

Cabe aclarar que para un perceptrón simple con unidades "threshold", el problema a resolver por la red debe ser *linealmente separable* para encontrar una solución por medio de la red, esto quiere decir que el problema a resolver debe existir plano puede ser encontrado en el espacio de las entradas, el cual separe las entradas cuya salida resulta en 1 y de las que resultan en -1.

Considerando problemas linealmente separable, se proporciona una regla de aprendizaje simple que consiste en modificar los pesos de la siguiente manera.

$$w_{ik}^{t+1} = w_{ik}^{t} + \delta w_{ik}$$

dado un patrón μ con una salida deseada ζ^{μ}

$$\Delta w_{ik} = \eta(1 - \zeta_i^{\mu} O_i^{\mu})\zeta_i^{\mu}\xi_k^{\mu} = \eta e \xi_k^{\mu}$$

donde η es la constante de aprendizaje y e es el error, es decir, la diferencia entre la salida deseada y la resultante de la red.

1.6. Resultados

A continuación, se muestran los resultados obtenidos mediante un perceptrón simple programado a través de un script de python.

Para la resolución se tomaron los siguientes valores para la AND, OR y XOR en caso de tener las dos entradas.

	ξ_2	ξ_1	O
μ_1	-1	-1	-1
μ_2	-1	1	1
μ_3	1	-1	1
μ_4	1	1	-1

	ξ_2	ξ_1	O
μ_1	-1	-1	-1
μ_2	-1	1	1
μ_3	1	-1	1
μ_4	1	1	1

	ξ_2	ξ_1	O
μ_1	-1	-1	-1
μ_2	-1	1	-1
μ_3	1	-1	-1
μ_4	1	1	1

Tabla 1: Tabla de patrones para la lógica AND,OR y XOR respectivamente.

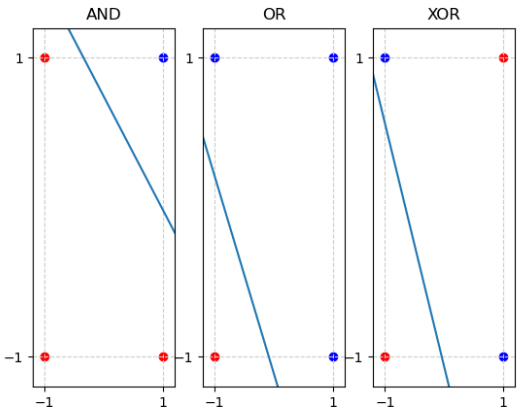
El perceptrón resolvió el problema y se obtuvo los siguientes resultados

```
 AND - 2 entradas
Encontrada la solución en  5  iteraciones
[[-0.65192656  0.66489019  0.87750002]]

 OR - 2 entradas
Encontrada la solución en  10  iteraciones
[[0.66691786 0.79198033 0.60092144]]

 XOR - 2 entradas
No se encontró ninguna solución.
```

(a) Resultado de los pesos sinápticos



(b) Rectas discriminatorias

Figura 1: Solución a los problemas de 2 entradas.

Podemos observar en la Figura [1a], que el algoritmo realizado para el perceptrón pudo encontrar una solución para la AND y para OR, en muy pocas iteraciones mientras que la XOR no pudo encontrarla en 2000 iteraciones, por eso es que no devolvió ninguna matriz con los valores de los pesos sinápticos.

Cabe destacar que el primer valor corresponde al umbral o bias impuesto a la salida de la red.

En la Figura [1b], se graficó la rectas discriminatorias para cada resultado encontrado y se puede observar con claridad que el problema de la XOR no es linealmente separable, ya que no existe ninguna recta capaz de separar el espacio de las entradas por su valor a la salida, mientras que la AND y la OR si.

Se repitió el experimento pero esta vez para las tablas AND y OR de 4 entradas, y se observó que demandó mayor cantidad de iteraciones para encontrar el resultado (Figura [2]). Esto tiene sentido ya que es mayor la cantidad de pesos a ajustar.

	ξ_4	ξ_3	ξ_2	ξ_1	O
μ_1	-1	-1	-1	-1	-1
μ_2	-1	-1	-1	1	-1
μ_3	-1	-1	1	-1	-1
μ_4	-1	-1	1	1	-1
μ_5	-1	1	-1	-1	-1
μ_6	-1	1	-1	1	-1
μ_7	-1	1	1	-1	-1
μ_8	1	1	1	1	-1
μ_9	1	-1	-1	-1	-1
μ_{10}	1	-1	-1	1	-1
μ_{11}	1	-1	1	-1	-1
μ_{12}	1	-1	1	1	-1
μ_{13}	1	1	-1	-1	-1
μ_{14}	1	1	-1	-1	-1
μ_{15}	1	1	-1	-1	-1
μ_{16}	1	1	-1	-1	1

	ξ_4	ξ_3	ξ_2	ξ_1	O
μ_1	-1	-1	-1	-1	-1
μ_2	-1	-1	-1	1	1
μ_3	-1	-1	1	-1	1
μ_4	-1	-1	1	1	1
μ_5	-1	1	-1	-1	1
μ_6	-1	1	-1	1	1
μ_7	-1	1	1	-1	1
μ_8	1	1	1	1	1
μ_9	1	-1	-1	-1	1
μ_{10}	1	-1	-1	1	1
μ_{11}	1	-1	1	-1	1
μ_{12}	1	-1	1	1	1
μ_{13}	1	1	-1	-1	1
μ_{14}	1	1	-1	-1	1
μ_{15}	1	1	-1	-1	1
μ_{16}	1	1	-1	-1	1

Tabla 2: Tabla de patrones para la lógica AND y OR respectivamente.

```
 AND - 4 entradas
Encontrada la solución en  34  iteraciones
[[-2.51915299  0.80016981  0.72052575  0.85470173  0.60640811]]

 OR - 4 entradas
Encontrada la solución en  86  iteraciones
[[3.93178762 0.80631556 2.23697182 0.80004826 0.86777089]]
```

Figura 2: Resultado de los pesos sinápticos para la resolución de los problemas

2. Ejercicio 2

2.1. Enunciado

2 Determine numéricamente cómo varía la capacidad del perceptrón simple en función del número de patrones enseñados.

2.2. Capacidad

La capacidad de almacenamiento de la red es un parámetro que nos indica cuantas imágenes podemos almacenar en una red de N neuronas, reconstruyendo con una probabilidad de error aceptada.

Para la resolución de este ejercicio se crearon problemas aleatorios, en donde se fue variando el tamaño del problema en cada iteración. A medida que aumentaba el tamaño, se intentaba resolver problemas de ese tamaño reiteradas veces para poder obtener el cálculo de cuantos problemas pudo resolver el perceptrón simple.

Aumentar el tamaño del problema significaba aumentar la cantidad de patrones que lo representan. Cada patrón posee una dimensión de 8 entradas, es decir, que la capa de entrada son 8 unidades. Y a cada uno se generó aleatoriamente se le asoció una salida con valor {-1,1} de manera azarosa.

2.3. Resultado

Para entradas de dimensión 8, se resolvieron 300 veces problemas generados aleatoriamente hasta alcanzar a problemas con 40 patrones diferentes.

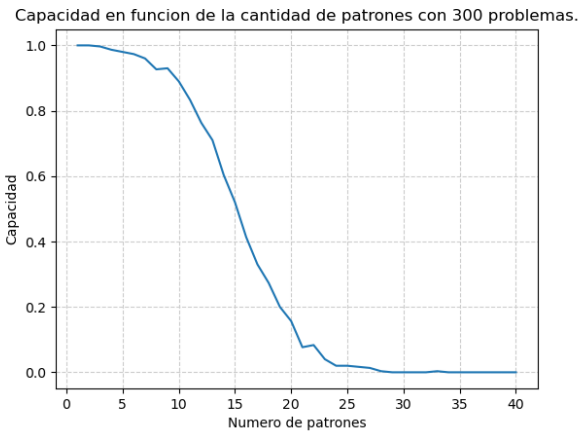


Figura 3: Capacidad de resolver problemas a medida que aumentan los patrones.

El resultado observado en la Figura [3] es lo esperado, ya que a medida que el número de patrones aumenta, es más difícil que el problema sea linealmente separable debido a la aleatoriedad de

los problemas generados y la probabilidad de que represente un problema no separable linealmente aumenta.

3. Ejercicio 3

3.1. Enunciado

3 *a*) Implemente un perceptrón multicapa que aprenda la función lógica XOR de 2 entradas y de 4 entradas (utilizando el algoritmo Backpropagation.)

b) Muestre cómo evoluciona el error durante el entrenamiento.

c) Para una red entrenada en la función XOR de dos entradas, grafique el error en función de dos pesos cualesquiera de la red. De ejemplos de mínimos locales y mesetas.

d) Idem (c) pero computando el error para cada patrón de entrada por separado.

3.2. Backdrop Propagation

Una de las formas de aprender, es realizar el gradiente descendiente, en donde modifiquemos los pesos sinápticos de tal manera que se reduzca el error que se presenta a la salida de la red. Existen varias formas de realizar gradiente descendiente, aunque al estar en presencia de una red feed-forward, al únicamente existir conexiones entre neuronas de diferentes capas se utiliza la técnica **backdrop propagation**.

Esta técnica consiste en propagar el error desde la salida hasta las entradas, calculando deltas que derivan de la derivada parcial del error respecto a los pesos que modificamos.

Las ecuaciones utilizadas para un perceptron de M capas, la modificación de los pesos fueron
Error a la salida $\delta_i^M = g^{'}(h_i^M)[\zeta_i^u - V_i^M]$
Error en capas intermedias $\delta_j^{m-1} = g^{'}(h_j^{m-1}) \sum_i w_i j^m \delta_i m$
Actualización de pesos $w_{ij}^{new} = w_{ij}^{old} - \Delta w_{ij}$

$$\Delta w_{ij} = \eta V_j^{m-1} - \delta_i^m$$

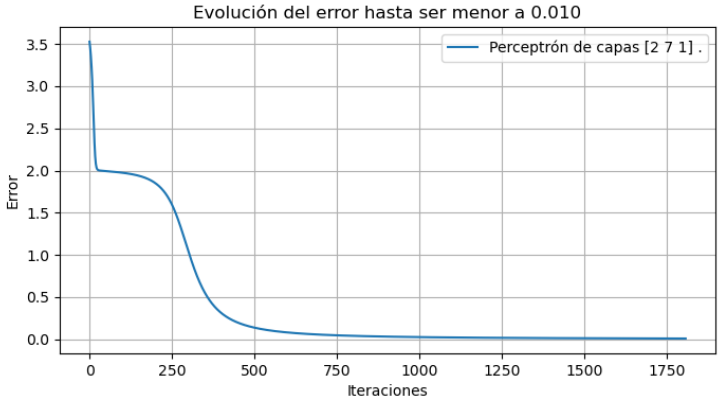
Para la función de activación se eligió $g(x) = tanh(x)$, cuya derivada es $g^{'}(x) = 1 - tanh^2(x)$

3.3. Resultados

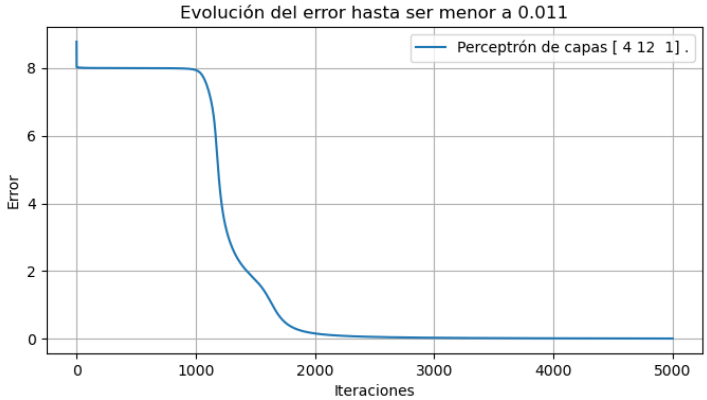
En la Figura [4a] podemos ver como ahora el XOR de dos entradas pudo llegar a una solución pasando las 1750 iteraciones con el perceptrón de dos capas con una capa intermedia de 7 neuronas. A pesar de esta vez resolverse, se puede observar como las iteraciones fueron relativamente mayor a los casos del perceptrón simple, lo cual es esperable ya que los pesos a ajustar los superan en cantidad.

También se puede observar de la Figura [4b] que el perceptrón pudo reducir considerablemente el error para el XOR de 4 entradas, aunque no logró reducir el error hasta 0,005 en 5000 iteraciones como fue propuesto, se considera que pudo resolver el problema planteado. Además se destaca las 1000 iteraciones que demoró el perceptrón en salir del error igual a 8. Algo parecido se había observado en la anterior figura, pero con un error menor y menos iteraciones.

Para reducir esta meseta generada, podría haberse adoptado una η más grande para las primeras iteraciones, de manera de que el error varíe más inestablemente hasta lograr salir de la meseta. Luego, con el correr de las iteraciones reducir el η para lograr un acondicionamiento de los pesos sinápticos más fino y lograr la mejor solución posible.



(a) XOR - 2 entradas



(b) XOR - 4 entradas

Figura 4: Evolución del error para el perceptrón multicapa.

En la Figura [5], es el resultado de variar dos pesos aleatoriamente elegidos de capas de pesos aleatoriamente elegidos de las matrices W encontradas en la resolución de la XOR de dos entradas. Estos pesos sorteados fueron $w_1 = w_{01}$ y $w_2 = w_{42}$ de la primera capa de pesos sináptico.

Se puede notar con facilidad que existe un mínimo local en donde esta ubicado respecto a estos dos pesos ya que no hay ninguna variación de los pesos elegidos tal que el error disminuya.

Cabe destacar existen diversas zonas donde el color no varía, por lo que podemos asegurar que son mesetas de error en donde en un rango de valores de w_1 y w_2 el error no se ve afectado por estos cambios. Estas mesetas se pueden ver principal mente en las cuatro esquinas.

La peor dirección que se puede tomar para variar dichos pesos, es aumentando el segundo peso y disminuyendo el primer peso elegido, de esta manera, se alcanzan hasta un error de valor que supera $\Delta E = 5$, valor que se encuentra nos aleja considerablemente del error deseado.

Gracias a la Figura [6], se observa como la variación de dichos pesos influyá en la figura anteriormente analizada. La mayoría tiene zonas en donde el error no varía apreciablemente y zonas en donde la diferencia de error alcanza valores que superan el 1,75. Esto es congruente con la Figura [5] ya que esta zona de mayor error se encuentran todos en lugares distintos, coincidiendo únicamente en la zona se encuentra el mayor error y la única región en la que no coincide es donde se encuentra el menor error de la misma figura.

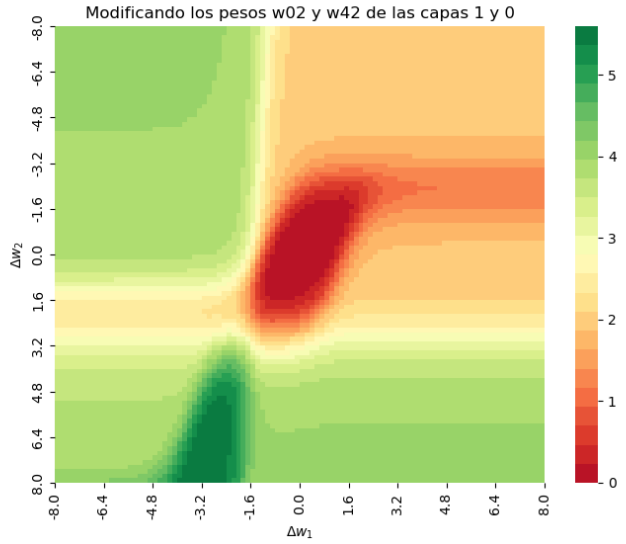


Figura 5: Mapa de calor de la diferencia de error generado al variar las w en dos posiciones.

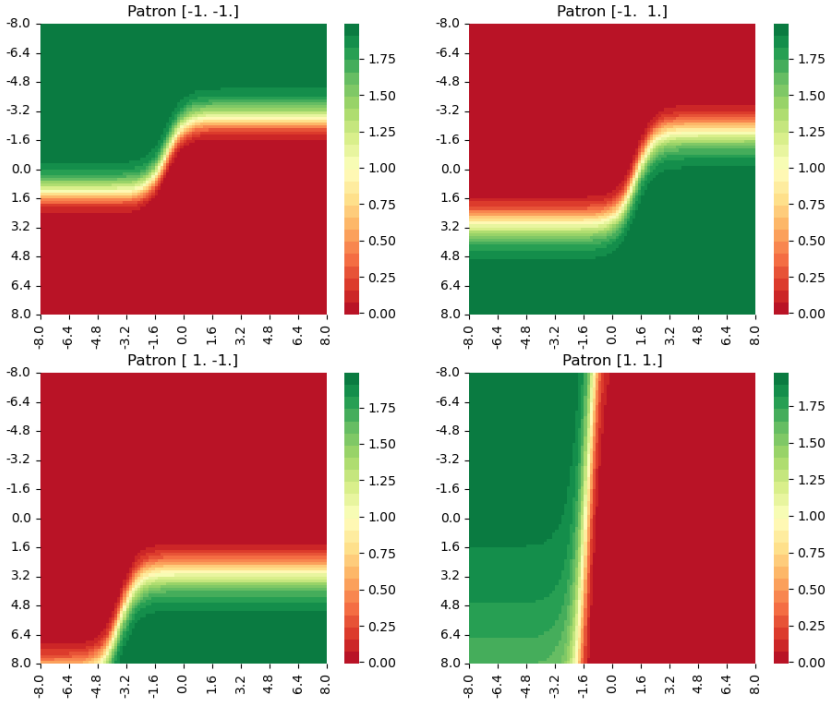


Figura 6: Mapa de calor de la diferencia de error a la salida generado en cada patrón al variar las w en dos posiciones.

4. Ejercicio 4

4.1. Enunciado

3 *a*) Implemente una red con aprendizaje Backpropagation que aprenda la siguiente función

$$f(x, y, z) = sin(x) + cos(y) + z$$

donde: x e y $\in [0, 2\pi]$ y z $\in [-1, 1]$. Para ello construya un conjunto de datos de entrenamiento y un conjunto de evaluación. Muestre el error en función de las epocas de entrenamiento.

b) ¿Como varía el número de iteraciones necesarias en función del tamaño del minibatch, y de la constante de aprendizaje? ¿Y el tiempo total de entrenamiento?

4.2. Resolución

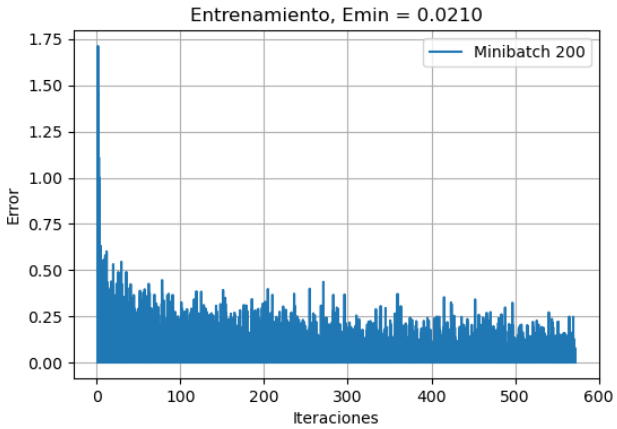
Por medio de la prueba y error continua, se logró converger a la solución con un error considerablemente bajo por medio de un perceptrón de dos capas, con 12 neuronas en la capa intermedia. Se realizaron todos los entrenamientos con 1000 muestras de entrenamiento y 300 muestras de testeo, generados de manera totalmente aleatoria.

Para el aprendizaje de la red, se eligió los vectores de entrenamiento pertenecientes al *minibatch* de manera aleatoria, y luego se calculó el error promedio sobre todo el set de testeo.

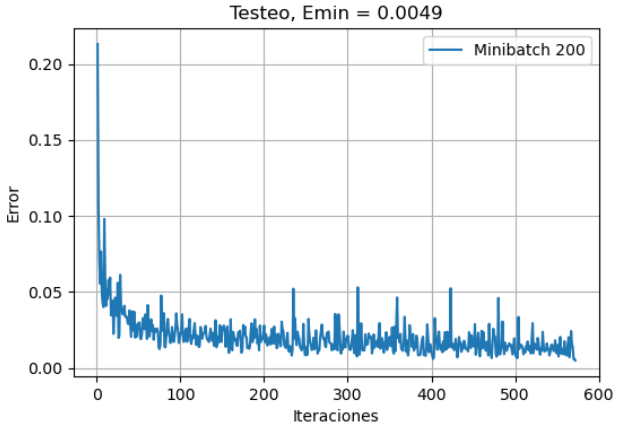
4.3. Resultado

En las siguientes Figuras se observan como a medida que el minibatch fue disminuyendo, la cantidad de epocas necesarias para alcanzar el error deseado disminuye considerablemente, aunque el error disminuye de manera menos pronunciada. Esto se puede explicar debido a que los ajustes realizados cuando los batch son pequeños, son más imprecisos debido a que poseen menos información que los batch de mayor tamaño.

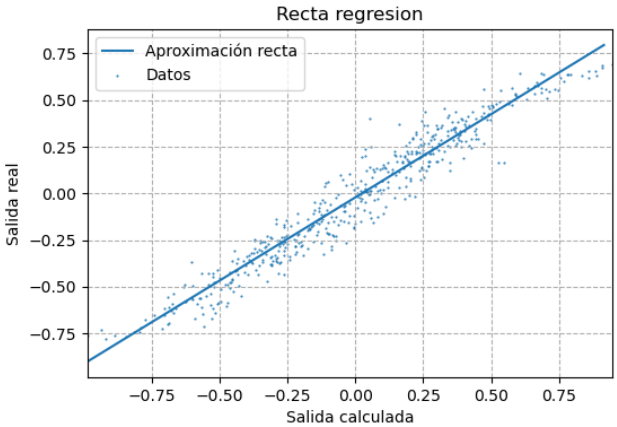
Además, el tiempo de ejecución del programa fue disminuyendo considerablemente, ya que los ajustes en la matriz W por medio de backdrop propagation se realizaban de manera más recurrente debido a que los batch eran de menor cantidad de vectores de entrenamiento.



(a) Error para los vectores de testeo

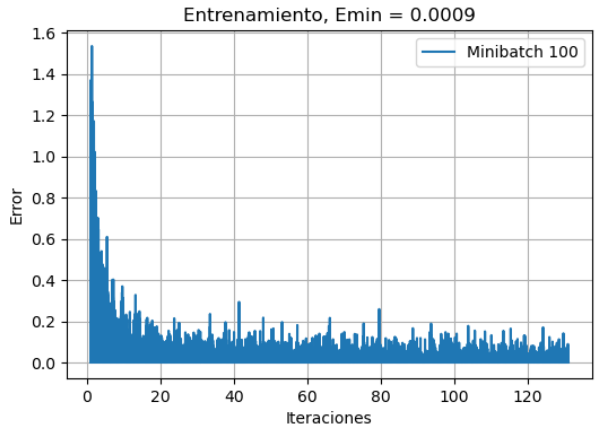


(b) Error para los vectores de testeo

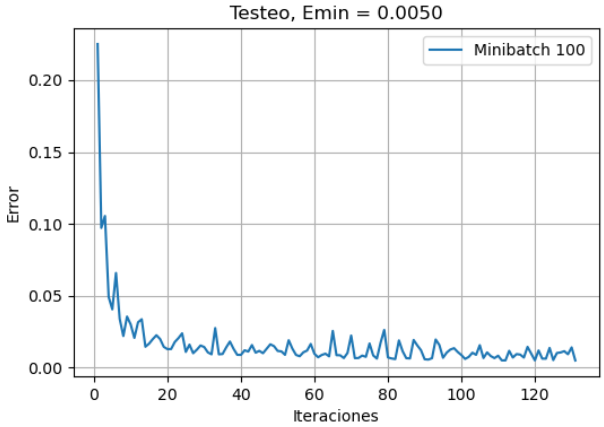


(c) Recta de regresión

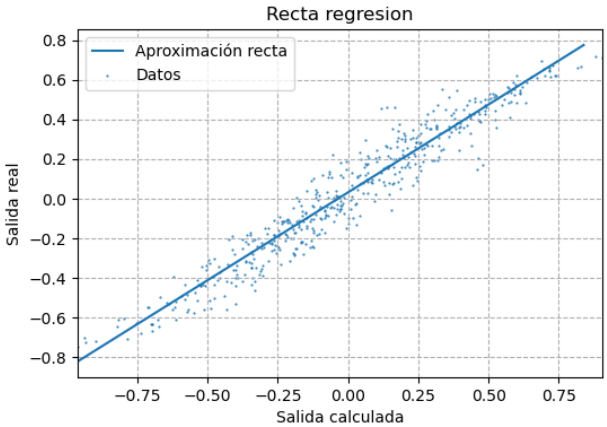
Figura 7: Minibatch de 200 patrones



(a) Error para los vectores de testeo

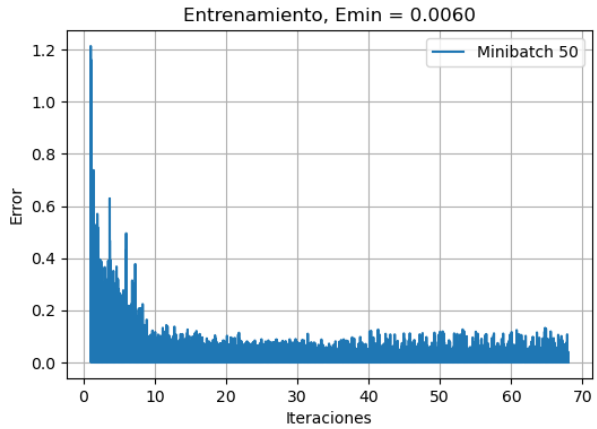


(b) Error para los vectores de testeo

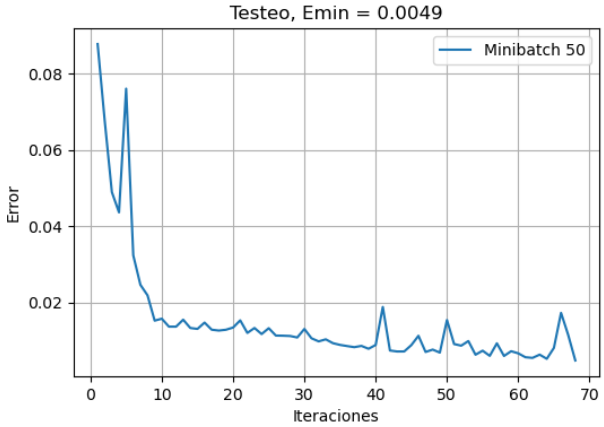


(c) Recta de regresión

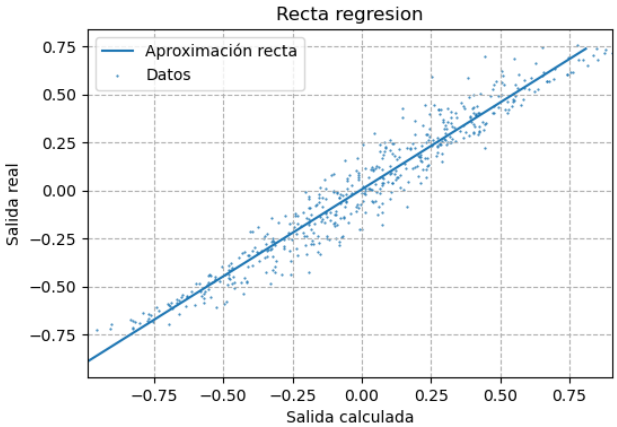
Figura 8: Minibatch de 100 patrones



(a) Error para los vectores de testeo



(b) Error para los vectores de testeo



(c) Recta de regresión

Figura 9: Minibatch de 50 patrones

5. Ejercicio 5

5.1. Enunciado

3 *a*) Encontrar un perceptrón multicapa que resuelva una XOR de 2 entradas con un algoritmo genético. Graficar el fitness a lo largo del proceso de evolución.

b) Cómo impacta en el aprendizaje la constante de mutación, la probabilidad de cross-over y el tamaño de la población.

5.2. Resolución

El algoritmo genético implementado consistió en 3 etapas esenciales, la primera es la reproducción de la población la cual tiene una probabilidad de ocurrir en función del fitness de cada individuo, y luego la cruza entre individuos de una misma población y la mutación de individuos, ambas con una probabilidad de ocurrencia arbitraria.

Para la función fitness se utilizó la ecuación $fitness = 1 - \frac{E}{4}$ donde E es el error provocado por un individuo en función de la salida real que genera.

Los individuos de una población fueron generados de manera totalmente aleatoria.

La probabilidad de reproducción de un individuo dependerá de que tan grande es el fitness del individuo respecto al de la población en general. Dicha probabilidad se cálcula mediante la siguiente ecuación.

$$\mathbf{P}(\text{reproducción}) = \frac{fitness_i}{\sum_{\forall j \in poblacion} fitness_j}$$

Para la cruza de individuos de una misma población se fue tomando de a pares, sin repetir, sorteando si dichos individuos se cruzaban con una probabilidad de cruza definida arbitrariamente. En caso de cruzarse, se realizó un intercambio de columnas para cada matriz de pesos sinapticos que poseía la red neuronal.

En el caso de la mutación, se agarró cada individuo, se sorteó la ocurrencia de la mutación con una probabilidad definida arbitrariamente y se le agregó un ruido gaussiano de media $\mu = 0$ y $\sigma^2 = 0{,}5$, es decir, $W_{mut} = W + N$, donde N es una matriz cuyos valores provienen de $\mathcal{N}(\mu, \sigma^2)$

La condición de corte elegida para nuestro sistema fue un Fitness promedio para la pobalción de un valor mayor a 0.9.

5.3. Resultados

Primero se realizó el experimento para una población de 100 individuos con una probabilidad de cruza y de mutación de 0.6. Se corrió dos veces el algoritmo genético pero en una de las corridas se consideró que el individuo más apto de la población siempre se reprodujera.

En la Figura [10] se puede observar como la implementación del vector elite no necesariamente mejora la rapidez con la que converge la población a la solución.

Luego se procedió a variar el tamaño de la población y las probabilidades definidas arbitrariamente y se graficó la cantidad de generaciones requeridas para cada una de las variantes para alcanzar el objetivo del fitness de la población. De esta manera podemos darnos una idea de la influencia de cada parametro a la solución.

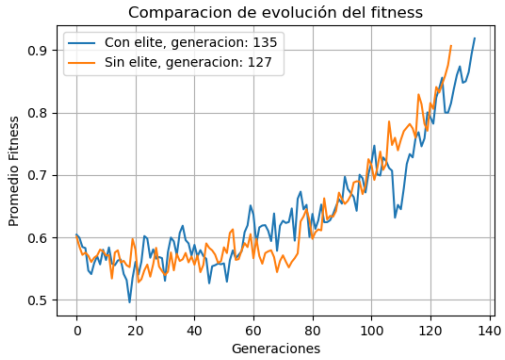
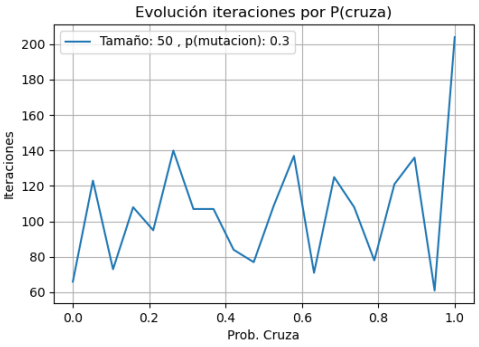
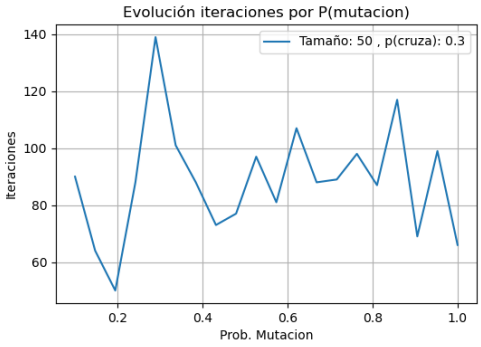


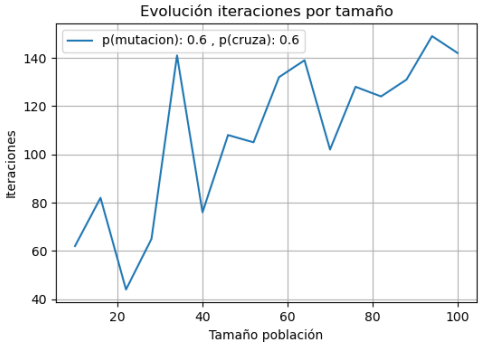
Figura 10: Comparación reproducción con y sin el vector Elite.



(a) Variando probabilidad de cruza de 0 a 1



(b) Variando probabilidad de mutación de 0.1 a 1



(c) Variando el tamaño de población de 10 a 100

Figura 11: Cantidad de generaciones según cada parámetro fijado arbitrariamente.

Para el caso del tamaño de la población, observamos en la Figura[11c] que la cantidad de generaciones necesitadas para lograr el objetivo fue aumentando a medida que la población aumentaba.

En cuanto a la probabilidad de mutación, no se puede extraer nada con demasiada contundencia ya que el pirco de mayor cantidad de generaciones ocurrió cercano al de menor cantidad y se observa una evolución estable; cabe destacar que en caso de haber ser la probabilidad 0, el sistema queda estancado ya que la población se ve poblada de un solo individuo y no logra encontrar nuevos individuos debido a que la cruza de estos (por como fue implementado) no lograba un individuo nuevo.

Por último, se observa en la Figura [11a] que la probabilidad de cruza tiene un comportamiento estable como se pudo observar con el de mutación, con la particularidad que en nuestra imple-

mentación, al de lo que pasaba con la mutación pero con probabilidad 0, cuando la probabilidad de cruza es 1 es cuando un peor rendimiento se encontró.

6. Ejercicio 6

6.1. Enunciado

3 Encontrar un perceptrón multicapa que resuelva una XOR de 2 entradas mediante simulated annealing. Graficar el error a lo largo del proceso de aprendizaje.

6.2. Simulated Annealing

Consiste en generar una variación en los pesos sinápticos de manera estocástica y aceptar si el cambio en el error es menor a cero con probabilidad 1 o si es mayor a cero con una probabilidad

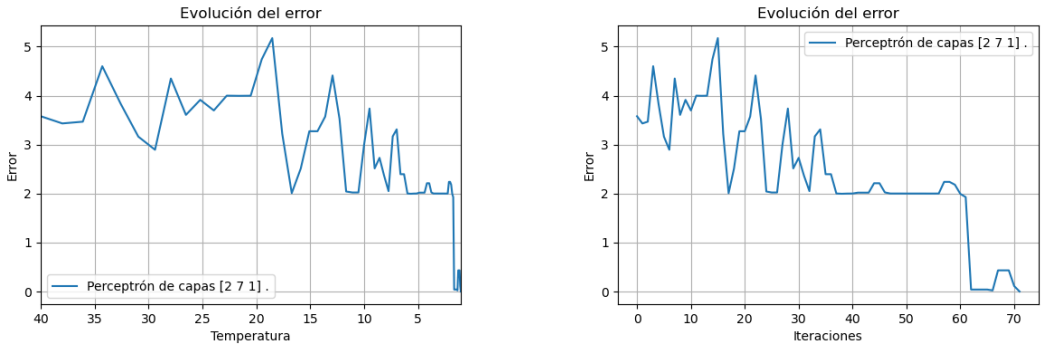
$$p = e^{-\frac{\Delta E(k)}{\beta T}}$$

Donde $\Delta E(k)$ es la variación del error, β es una constante y T es la temperatura. A su vez, se sigue un protocolo de enfriamiento $T(k+1) = \alpha T(k)$ con $\alpha \in [0, 1]$. Para este ejercicio se eligió un $\alpha = 0,9$ y un $\beta = 1$.

Estos algoritmos permiten movimientos ascendentes en cuanto al error y así evitar quedar atrapado prematuramente en un mínimo local.

6.3. Resultados

Con una temperatura inicial de $T = 40$, se lanzó el experimento y se graficó la evolución del error a medida que avanzaban las iteraciones y se modificaba la temperatura. Ambas Figuras muestran que los saltos de error ascendentes se dan cuando la temperatura es mayor y esto es debido a la probabilidad propuesta en por el *simulated annealing*. A partir de $T = 5$ vemos que el sistema se enfrió lo suficiente para aceptar pocos cambios en la matriz que aumenten el error, y luego para temperaturas cercanas a 0 convergió rápidamente a un óptimo. Ambas figuras evolucionan similarmente, ya que a medida que la iteración aumentaba, la temperatura disminuía.



(a) En función a la temperatura.

(b) En función a las iteraciones.

Figura 12: Error a lo largo de la evolución del sistema.