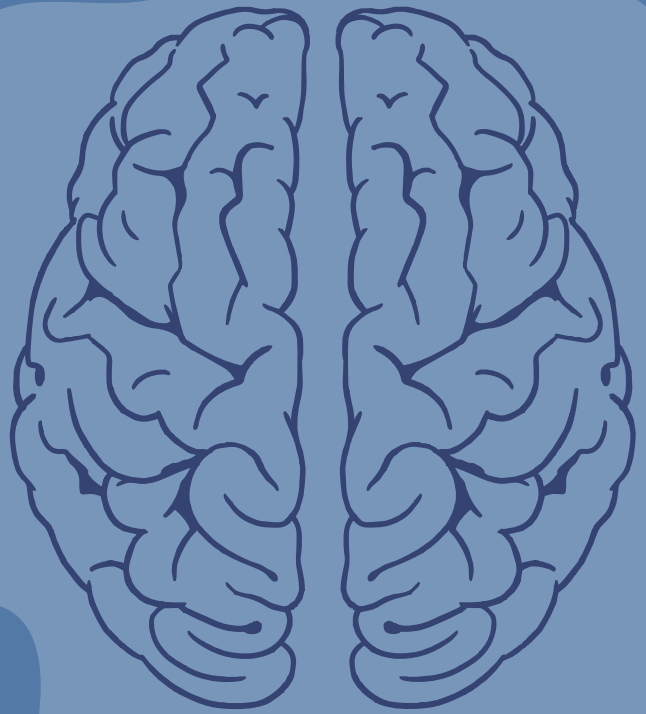
Redes Neuronales

SIA - ITBA - 2021



INTRODUCCIÓN

El siguiente trabajo práctico fue implementado en Python. Toda la configuración se realiza desde un archivo yaml, por defecto config.yaml.

Para realizar los ejercicios pedidos se tuvieron que implementar distintos tipos de perceptrones. Lo que caracteriza a cada una de estos es la función de activación utilizada a la hora de realizar la predicción de los valores y la forma de actualizar los pesos con los cuales se pondera el valor de cada entrada recibida.

A su vez, se tuvo que trabajar con Redes neuronales, las cuales funcionan como múltiples capas de perceptrones que trabajan conjuntamente para obtener su predicción.

01

Perceptrón Escalón

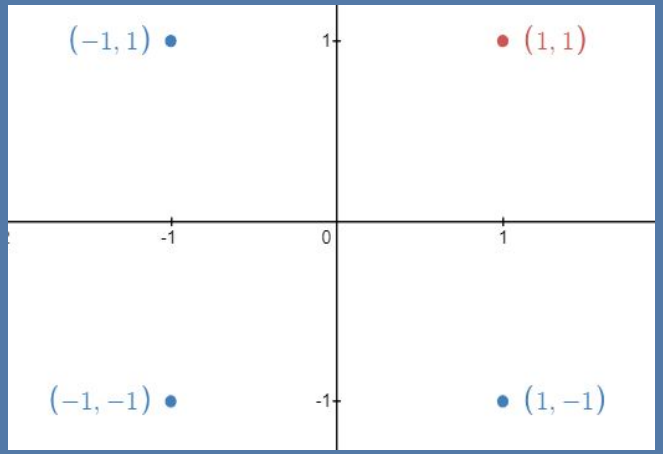
Ejercicio 1

Gracias al modelo de McCulloch y Pitts tenemos el primer acercamiento a nuestro modelo de perceptrón, donde la función de activación utilizada será la función signo que se adaptará al problema de separación lineal que presenta este ejercicio.

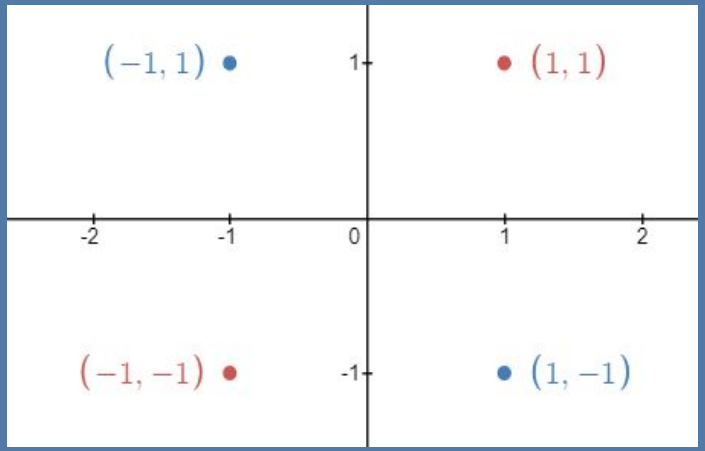
Sin embargo, este es solo el primer paso, para resolver nuestro problema. También utilizaremos el modelo de Rosenblatt para el cual deberemos asumir la conjetura de Hebb, que indica que las sinapsis de neuronas con estado de actividad correlacionados a lo largo del tiempo se refuerzan.

Con todo esto, tenemos el algoritmo que permite que nuestro perceptrón “aprenda”. Así podremos darle un conjunto de entrenamiento, en este caso los distintos puntos a separar y su valor esperado. Para la selección de los pesos iniciales se tomó un conjunto de pesos aleatoriamente seleccionados de manera uniforme entre -1 y 1

Datasets a analizar

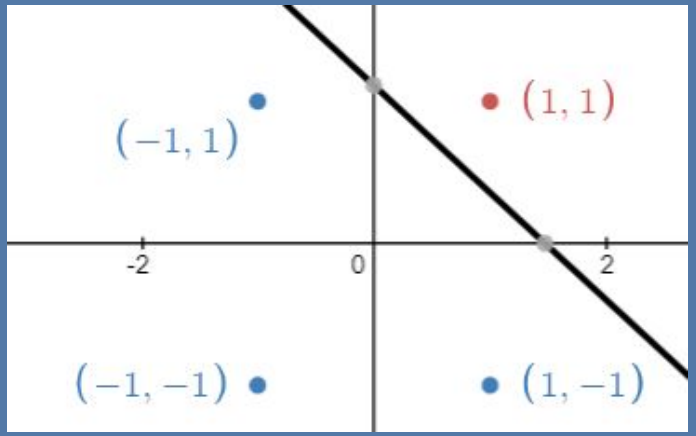


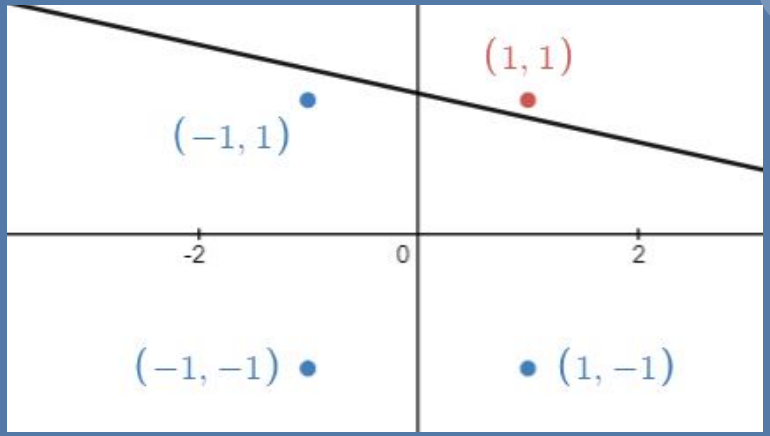
AND



XOR

Resultados para AND



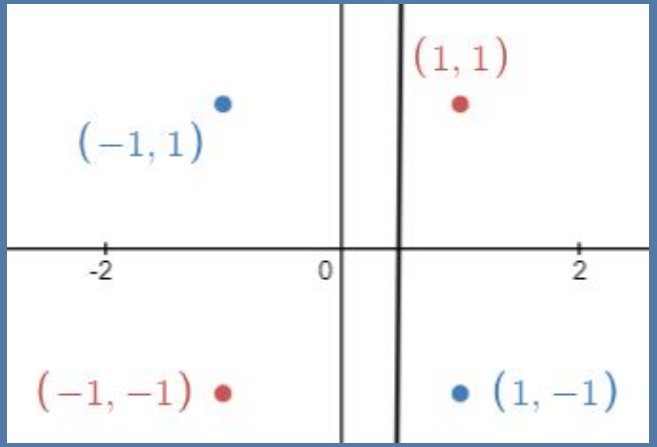


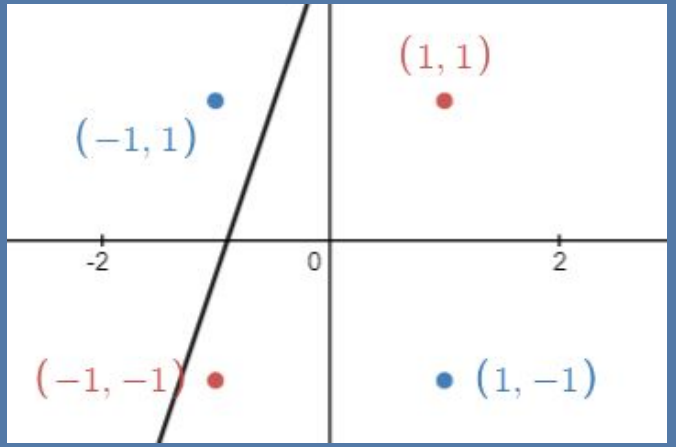
Caso AND

Como podemos comprobar, el perceptrón lineal encuentra una recta que soluciona el problema de separación lineal.

En los gráficos se muestra cómo distintas ejecuciones del algoritmo, encuentra distintas rectas. Esto se debe a la inicialización de los pesos aleatoria. A pesar de que las rectas sean distintas, el perceptrón siempre encuentra una que logra separar los puntos rojos de los azules, lo que nos lleva a concluir que el conjunto es linealmente separable.

Resultados para XOR





Caso XOR

A diferencia del caso anterior, el perceptrón no logra encontrar una recta para separar linealmente los puntos.

Esto se debe a que tal recta, como se puede comprobar a simple vista, no existe. En este caso, se concluye que el conjunto no es linealmente separable.

02

Perceptrón Lineal y No lineal

Ejercicio 2

Para este ejercicio se utilizaron otros dos tipos de perceptrones los cuales, si bien el modelo utilizado es igual al del perceptrón escalón, su función de activación y cálculo de la actualización de los pesos difiere.

El primero, el perceptrón lineal, en vez de utilizar una función escalonada como la función signo, usa la identidad. Esto permitirá que la salida esperada no deba pertenecer a conjuntos como {-1, 1} o {0, 1}, sino que podrá pertenecer a los reales.

El segundo, el perceptrón no lineal, utilizará funciones sigmoideas como la tangente hiperbólica o la función logística. Además, a la hora de calcular la actualización de los pesos, se multiplica por la derivada de la activación.

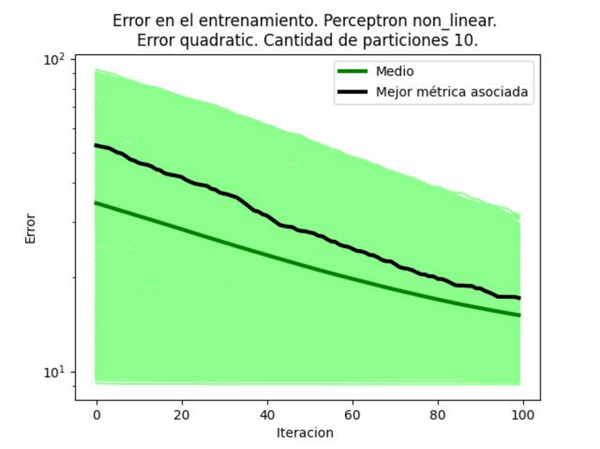
Los valores de entrada son tuplas de tres valores reales cuyos valores de salida es un número real. Dado que estos números estaban por fuera del intervalo [-1, 1], se decidió normalizar tanto los valores de entrada y de salida, dividiéndolos por 100, para que los resultados del perceptrón no lineal que está acotado, sean comparables con los del perceptrón lineal.

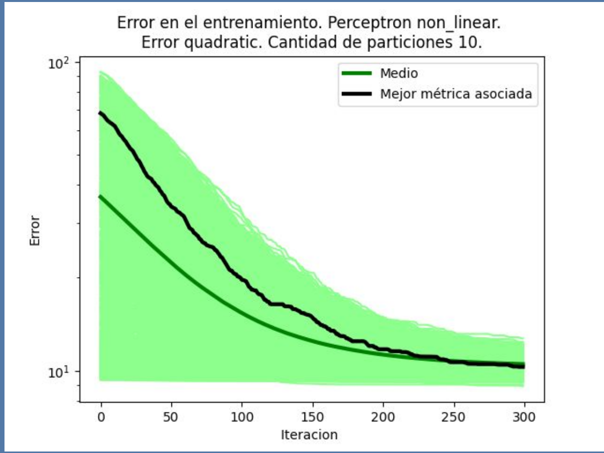
Ejercicio 2

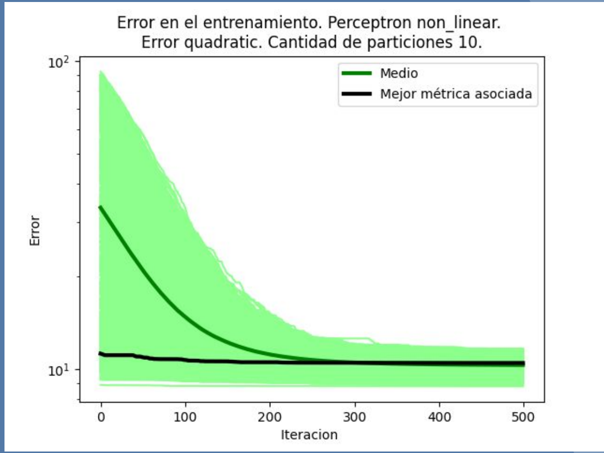
Este es un problema de aproximación, es decir que se busca encontrar una función (lineal o no lineal) que se aproxime lo máximo posible a los valores provistos.

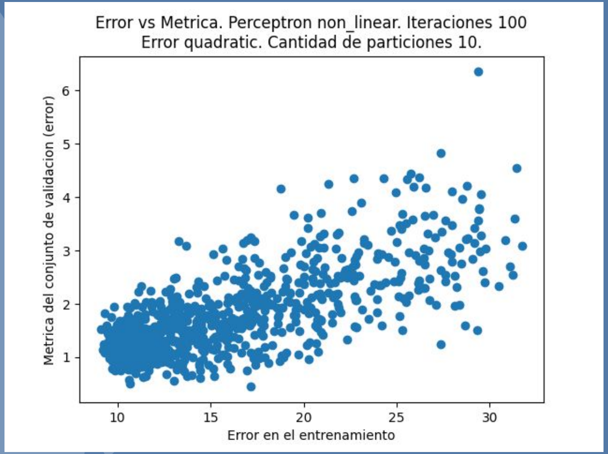
Para poder determinar cuan bien se está generalizando esta función, es decir, cuán bien estamos aprendiendo sus características fundamentales, vamos a utilizar el método de **cross validation** tomando la métrica error.

Esto nos permite determinar el mejor conjunto de prueba a elegir, y encontrar cual es el error que nuestra red neuronal tiene en un conjunto de puntos nuevo.

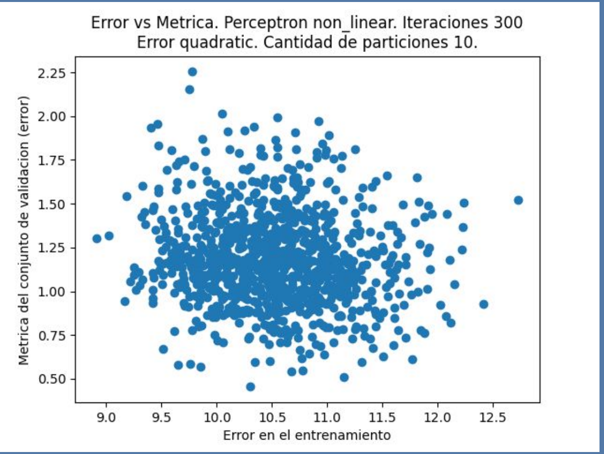




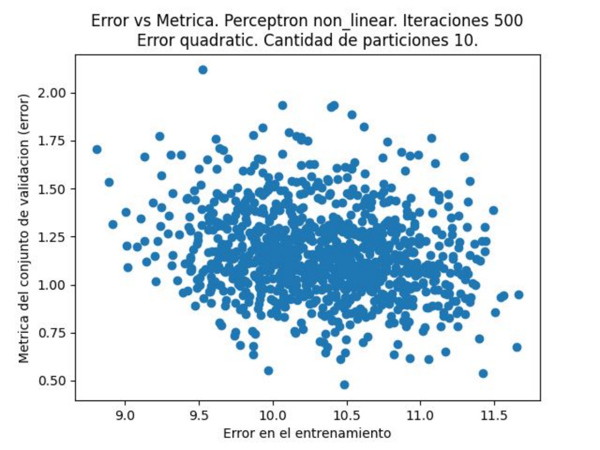




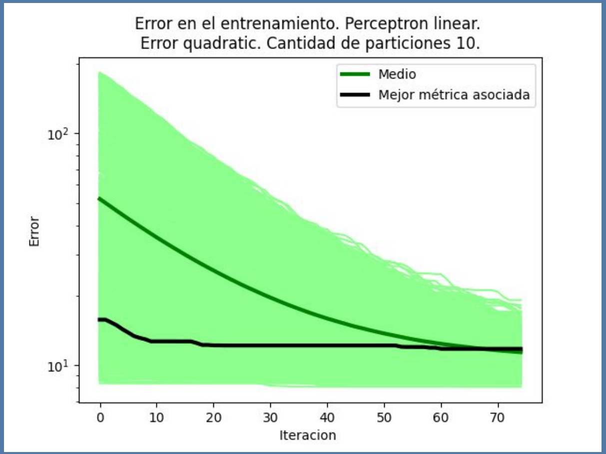
STD: 0.753
Mean: 1.682
Best: 0.447

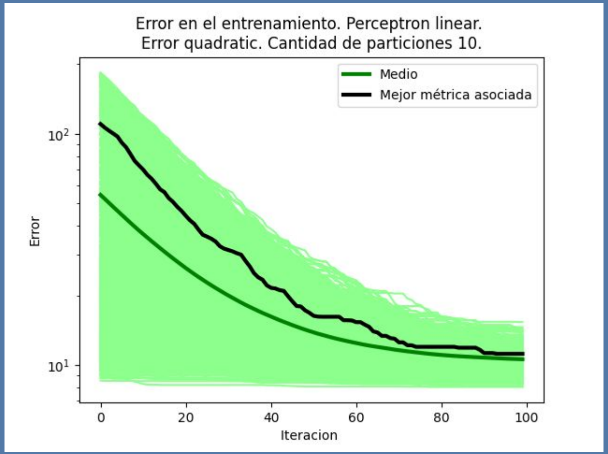


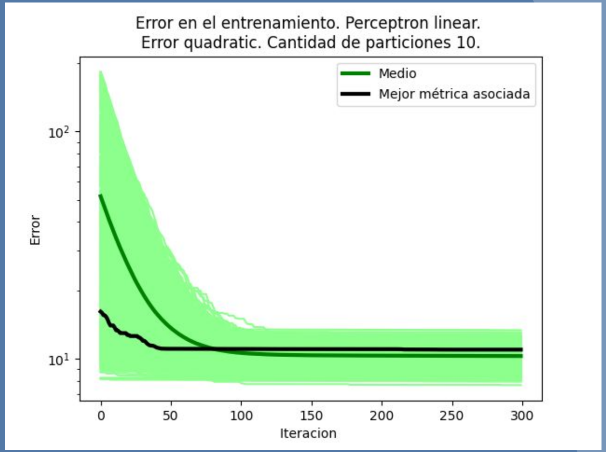
STD 0.263
Mean 1.182
Best: 0.454

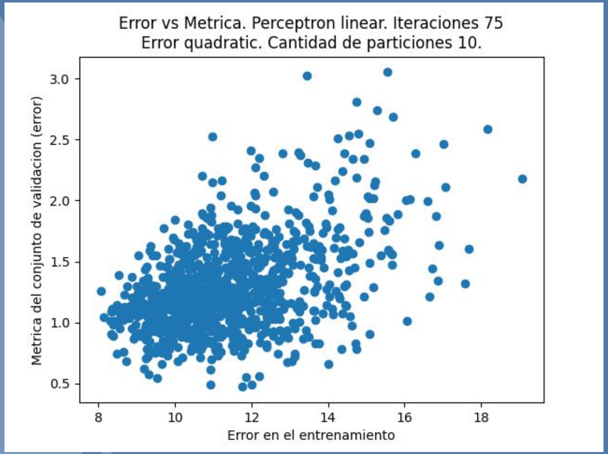


STD: 0.229
Mean: 1.161
Best: 0.478

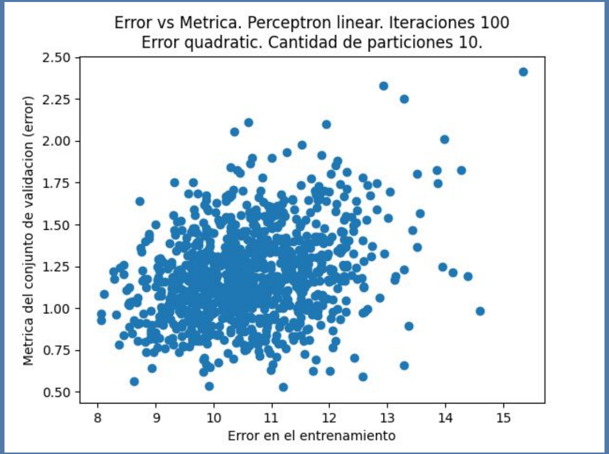




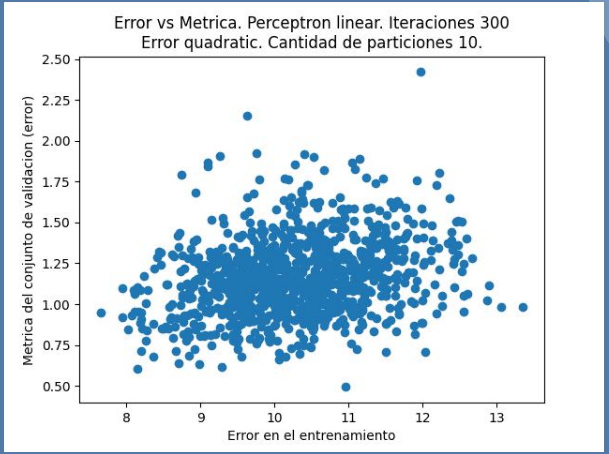




STD 0.366
Mean 1.287
Best: 0.472



STD 0.269
Mean 1.197
Best: 0.528



STD: 0.235
Mean: 1.168
Best: 0.492

Overfitting?

No lineal vs Lineal

Más iteraciones mejora el resultado → No se alcanzó overfitting

Tanto lineal como no lineal alcanzan buenos resultados → Problema lineal

A medida que el error de entrenamiento converge se achica el desvío estándar de las predicciones

Una vez que convergio el error de entrenamiento no hay una clara correlación entre una buena métrica y un buen entrenamiento

Una pregunta que sigue pendiente es:
¿Cómo podemos separar conjuntos no linealmente separables ?

03

Perceptrón multicapa

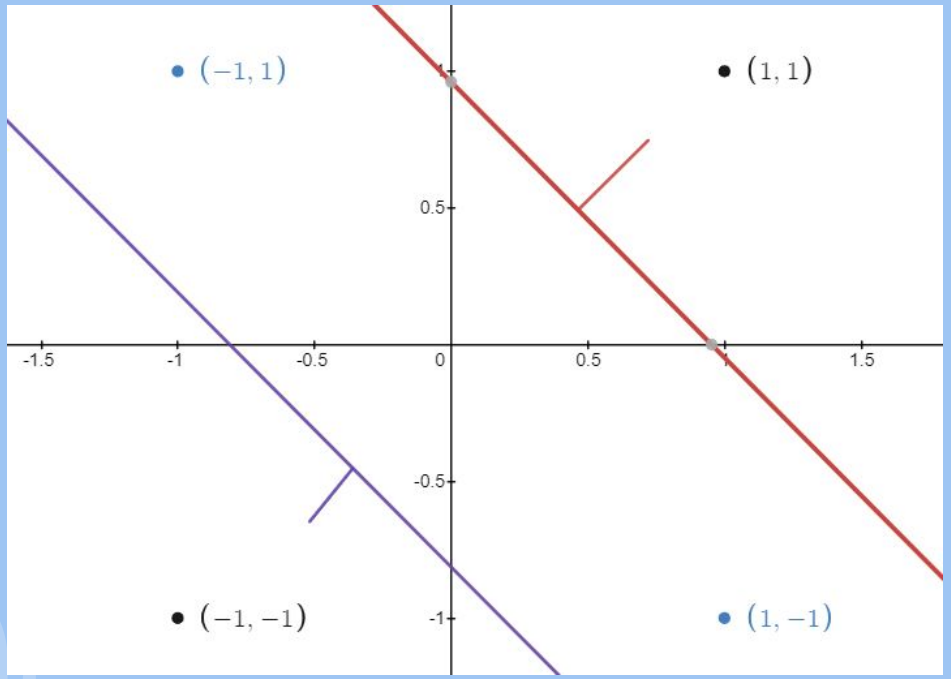
We need some backup!

Ejercicio 3

Para realizar este ejercicio, se utilizó el perceptrón multicapa, el cual no es más que un conjunto de perceptrones como los analizados hasta ahora, pero que trabajan en conjunto, divididos en distintas capas. Todas las capas intermedias de la red serán llamadas **capas ocultas**. El problema que surge ahora es de qué manera actualizar los pesos de los nodos en estas capas intermedias ya que no hay forma de compararlas con el valor esperado. Para esto se utiliza la técnica de 'retro propagar' la diferencia entre la predicción y el resultado de cada unidad de la capa de salida hacia las capas inferiores.

Una vez resuelto esto, habrá que considerar cómo actualizar los pesos. Para esto habrán dos opciones de manera secuencial o por lotes. Consideramos que la primera forma se asemejaba más al aprendizaje de una verdadera red neuronal.

XOR con 2 perceptrones de entrada y 1 de salida

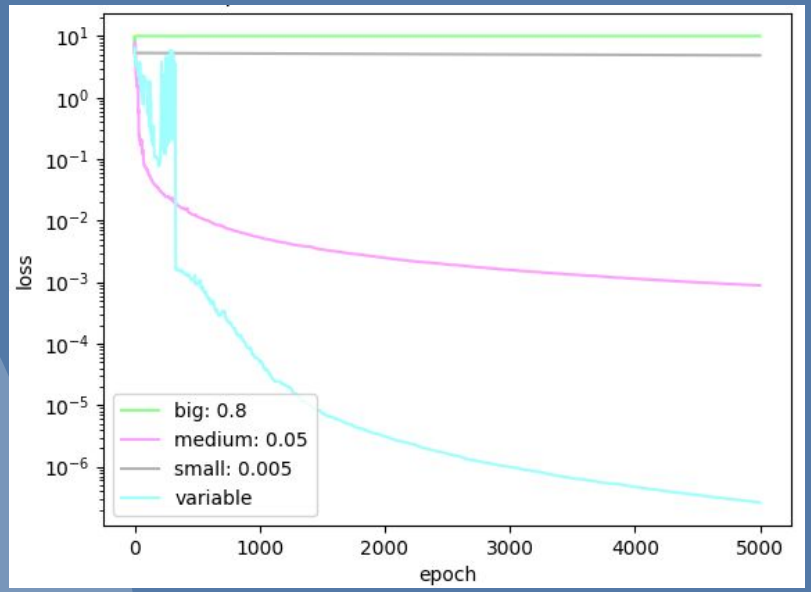


Ecuación de la recta: $\frac{-(w_1 x + w_0)}{w_2}$

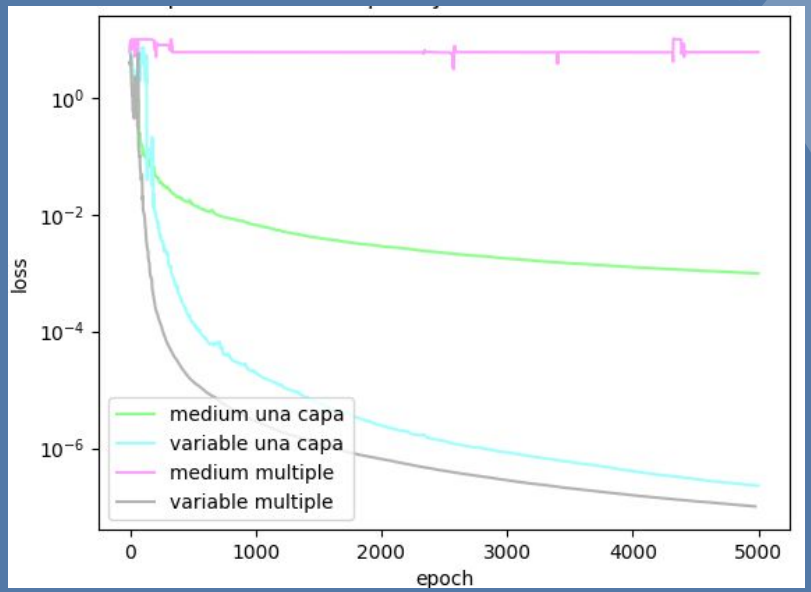
	(w_0, w_1, w_2)
P1 (rojo):	(-2.445, 2.565, 2.543)
P2 (violeta):	(-1.573, -1.944, -1.935)
P (salida):	(-2.583, -3.097, -3.143)

(-1, 1) → P1 = -0.902, P2 = -0.734 → P = tgh(0.6 * 2,517) > 0

Análisis de la tasa de aprendizaje a tomar



Último error de la red en base a la tasa de aprendizaje perceptrón multicapa [10 1]



Último error de la red en base a la tasa de aprendizaje entre red de 2 capas [10 1] y otra de múltiples [10, 7, 6, 1]

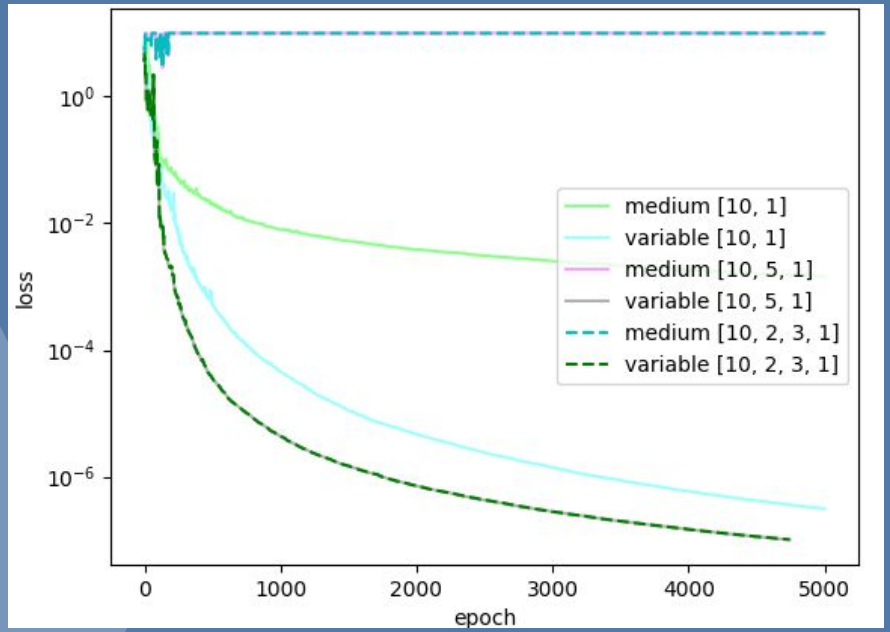
Caso Numeros pares e impares (Tasa de aprendizaje)

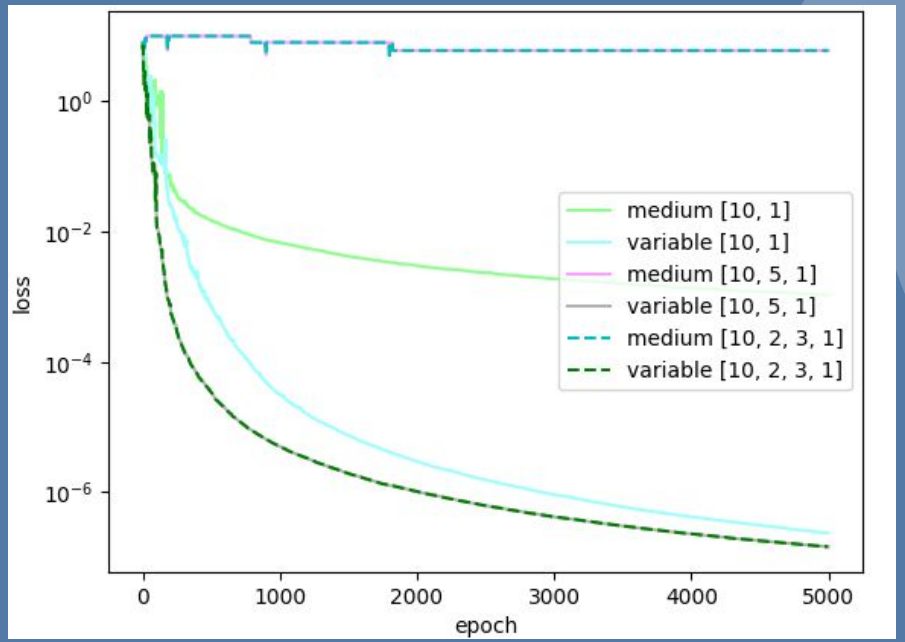
En el primer gráfico puede comprobarse como la tasa de aprendizaje small y big se estancan. El primer caso se debe a que el cambio termina siendo insignificante mientras que el segundo todo lo contrario, es demasiado grande por lo que termina divergiendo.

El caso de la tasa medium logra avanzar de manera constante sin divergir ni terminar de converger.

Por último, el variable comienza teniendo ruido, pero una vez se estabiliza, disminuye muy rápidamente, dado que este al estar mejorando constantemente, permite subir la tasa en esa dirección obteniendo así los mejores resultados.

Analisis cantidad de capas a tomar





Usando las mejores tasas de aprendizaje, errores de distintos modelos de perceptrones

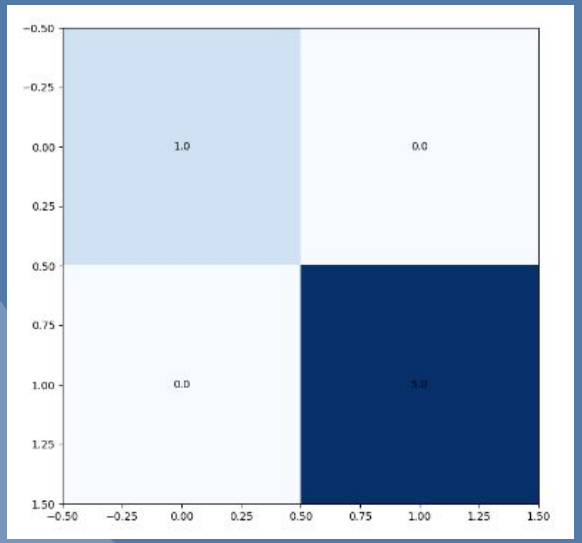
Caso Numeros pares e impares (Cantidad de capas)

En el gráfico se ve como el modelo con más nodos funciona mejor que el que tiene menos. Sin embargo, la forma en la que están subdivididos en capas pareciera no afectar dado que los resultados para el modelo [10, 5, 1] y [10, 2, 3, 1] terminan idénticos. En ambos con tasa fija se diverge mientras que con tasa variable logran obtener buenos resultados.

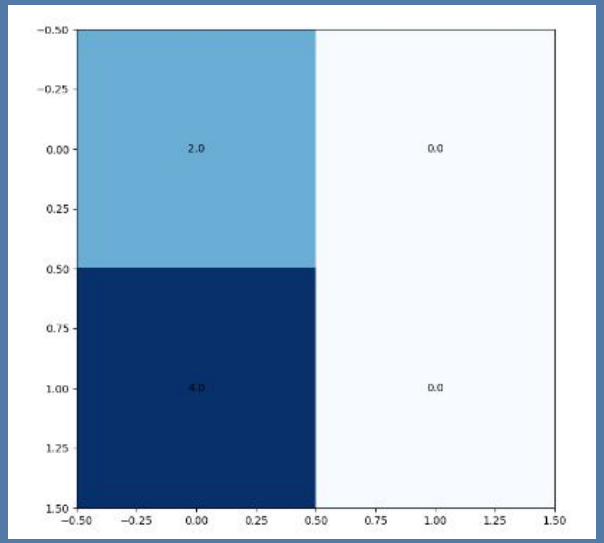
Por último, para el modelo [10, 1] se obtienen peores resultados, pero tanto la tasa media como la variable logra descender constantemente.

Consideramos que mayor cantidad de nodos puede ser mejor dado que permite adaptarse más al problema trazando mayor cantidad de rectas.

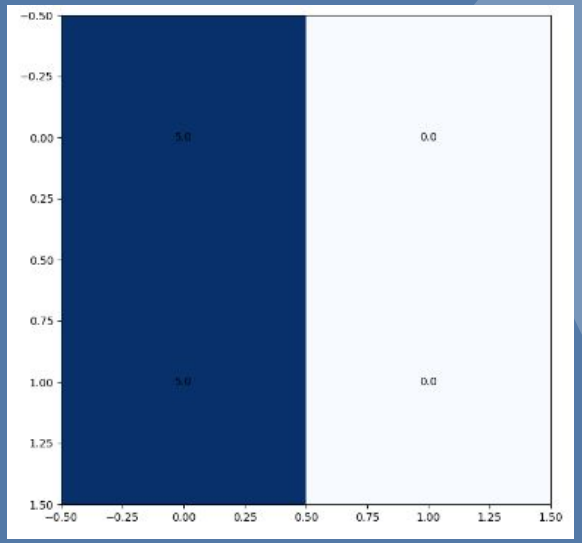
Resultados con k = 4 y 8 iteraciones



Resultados de red con menor error sobre puntos de entrenamiento



Resultados de red con mejor métrica sobre puntos de entrenamiento

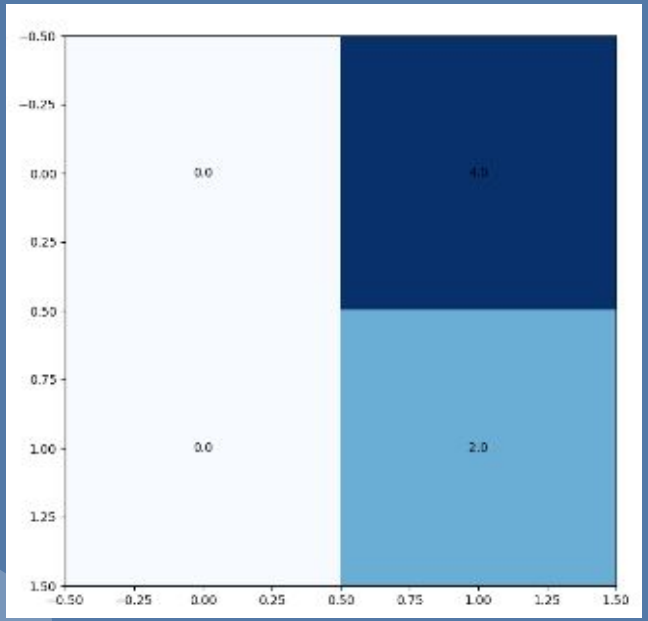


Resultados de red con mejor métrica sobre todos los puntos

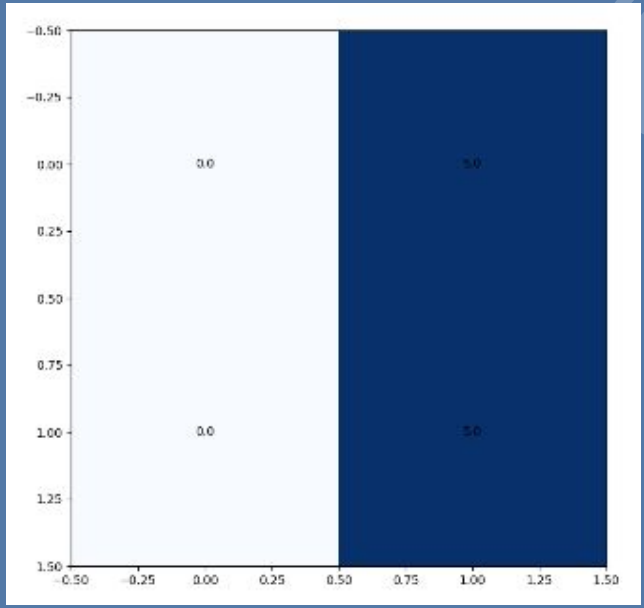
Caso Numeros pares e impares (conjunto seleccionado)

En este caso el primer gráfico muestra una red que tuvo overfitting, ya que fue infalible para el conjunto de entrenamiento. Sin embargo, la del segundo gráfico, fue muy mala para estos puntos pero logró, como se muestra en el tercer gráfico, generalizar mucho mejor y conseguir una métrica alta por lo que fue elegida.
Así, se puede comprobar que el método de validaciones cruzadas consigue el conjunto que permite mejor generalización.

Resultados con k = 4 y 15 iteraciones



Resultados de red con mejor métrica sobre puntos de entrenamiento



Resultados de red con mejor métrica sobre todos los puntos

```
Best accuracy = 0.75 Mean = 0.45833333333333333 Standard dev= 0.18352262954621038
```

Caso Numeros pares e impares (capacidad de generalizar)

Como puede comprobarse con los gráficos y mediante múltiples testeos, si bien la red elegida logró una métrica superior a la media, esta es de aproximadamente **0.5** lo cual tiene sentido dado que es la probabilidad de 'lanzar una moneda' o adivinar que tiene la red para cualquier valor. Esto nos confirma como, la red no está logrando generalizar y únicamente está devolviendo un valor al azar. De esta forma, el que haya una métrica mayor a la media es simplemente un golpe de suerte, que se confirma con un desvío estándar bajo.

Template de la presentacion de
https://slidesgo.co