Práctica 12: red neuronal

Raul L.

18 de mayo de 2022

1. Introducción

La última práctica es una demostración básica de aprendizaje a máquina: vamos a reconocer dígitos de imágenes pequeñas en blanco y negro con una red neuronal. El elemento básico de una red neuronal es un perceptrón que esencialmente es un híperplano (una línea si nos limitamos a dos dimensiones) que busca colocarse en la frontera que separa las entradas verdaderas y las entradas falsas. La dimensión d del perceptrón es el largo del vector x que toma como entrada, y su estado interno se representa con otro vector$w$ que contiene sus pesos. Para responder a una salida proporcionada a ello, el perceptrón calcula el producto interno de xw, es decir

$$\sum_{i=1} = x_i w_i$$

, y si esta suma es positiva, la salida del perceptrón es verdad, en otro caso es falso [1]. Para agarrar la onda con los perceptrónes, haremos primero uno más sencillo cuyo jale es identificar si$x > y$ para puntos en dos dimensiones, ya que así es fácil para nosotros visualizar lo que le pasa al perceptrón durante el entrenamiento[2].

2. Objetivo

Estudia de manera sistemática el desempeño de la red neuronal en términos de su puntaje F (F-score en inglés) para los diez dígitos en función de las tres probabilidades asignadas a la generación de los dígitos (ngb), variando a las tres en un experimento factorial adecuado[2].

3. Código

Para este código se utilizó como base el código de la doctora.

Código en Python `https://github.com/satuelisa/Simulation/blob/master/NeuralNetwork/ann.p`

Código creado en Python

`https://github.com/Raullr28/Resultados/blob/main/P_12`

```
n=0.99
g=0.01
b=0.50

fact_des= itertools.product((n,g,b),(n,g,b),(n,g,b))
factor=[]
factor_lab=[]
for i in fact_des:
    factor.append(i)
    factor_lab.append(str(i))

CV=[]
for n, g, b in factor:
    print('###############',n,g,b,'###############')
    replicas=[]
    repeticiones=10
    for rep in range(repeticiones):
        modelos = pd.read_csv('digits.txt', sep=' ', header = None)
        modelos = modelos.replace({'n': n, 'g': g, 'b': b})
```

Código 1: Representación de la función factorial.

```
  c = pd.DataFrame(contadores)
        c.columns = [str(i) for i in range(k)] + ['NA']
        c.index = [str(i) for i in range(k)]

        diagonal=[]
        for d in range(0,k):
            num=c.iloc[d][d]
            diagonal.append(num)

        FP=[]
        for e in c.columns[0:-1]:
            suma=sum(c[e])
            diag=diagonal[int(e)]
            FP.append(suma-diag)

        FN=[f for f in c['NA']]

        PCS= sum(diagonal)/(sum(diagonal)+sum(FP))
        RCL=sum(diagonal)/(sum(diagonal)+sum(FN))
        F_score= 2*(PCS*RCL)/(PCS+RCL)
        replicas.append(F_score)
```

Código 2: Representación FC.

4. Resultados

En una gráfica de violines podemos ver el comportamiento al variar el experimento factorial dando como resultado la mejor conbinacion para obtener el mejor puntaje .

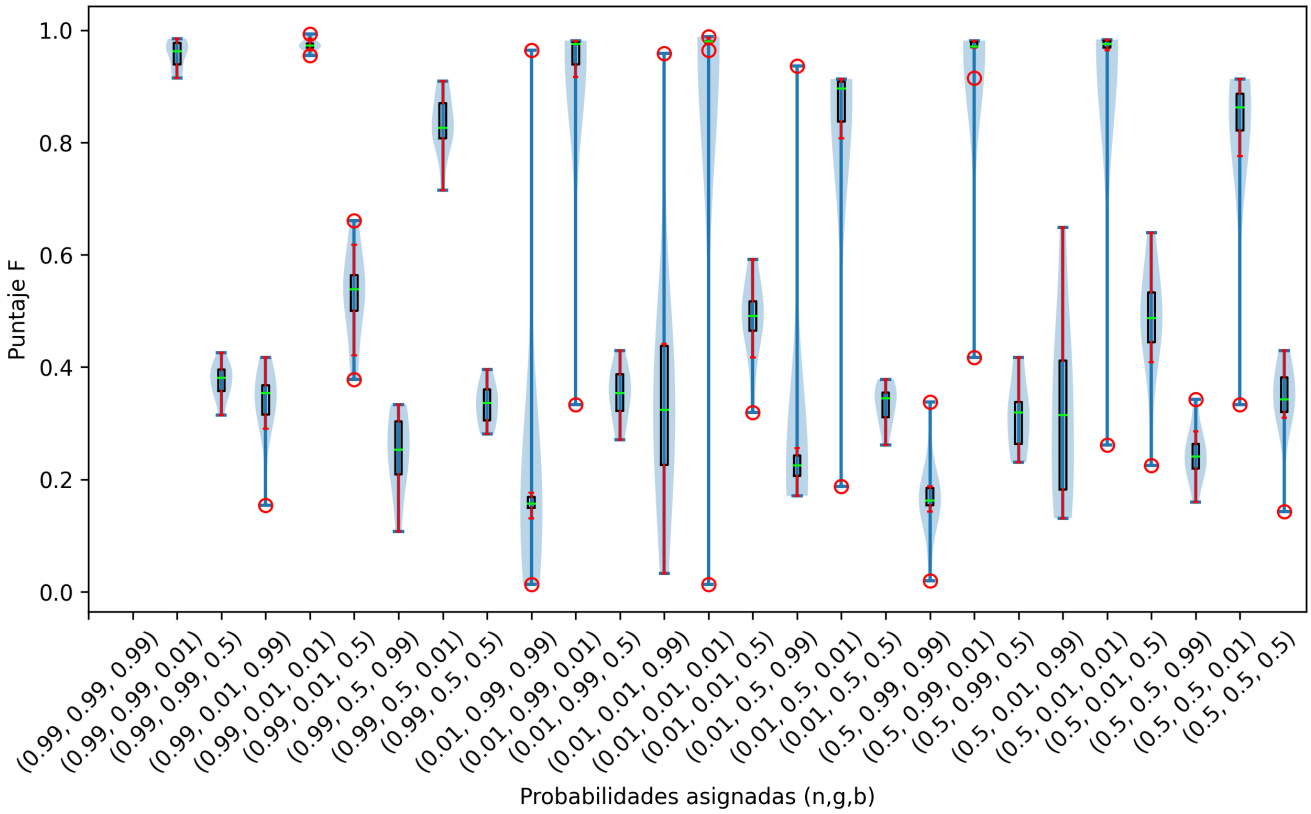

Figura 1: Gráfica de comparación.

5. Reto 1

Como un primer reto, extiende y entrena la red neuronal para que reconozca además por lo menos doce símbolos ASCII adicionales, aumentando la resolución de las imágenes a 5×7 de lo original de 3×5(modificando las plantillas de los dígitos acorde a este cambio).

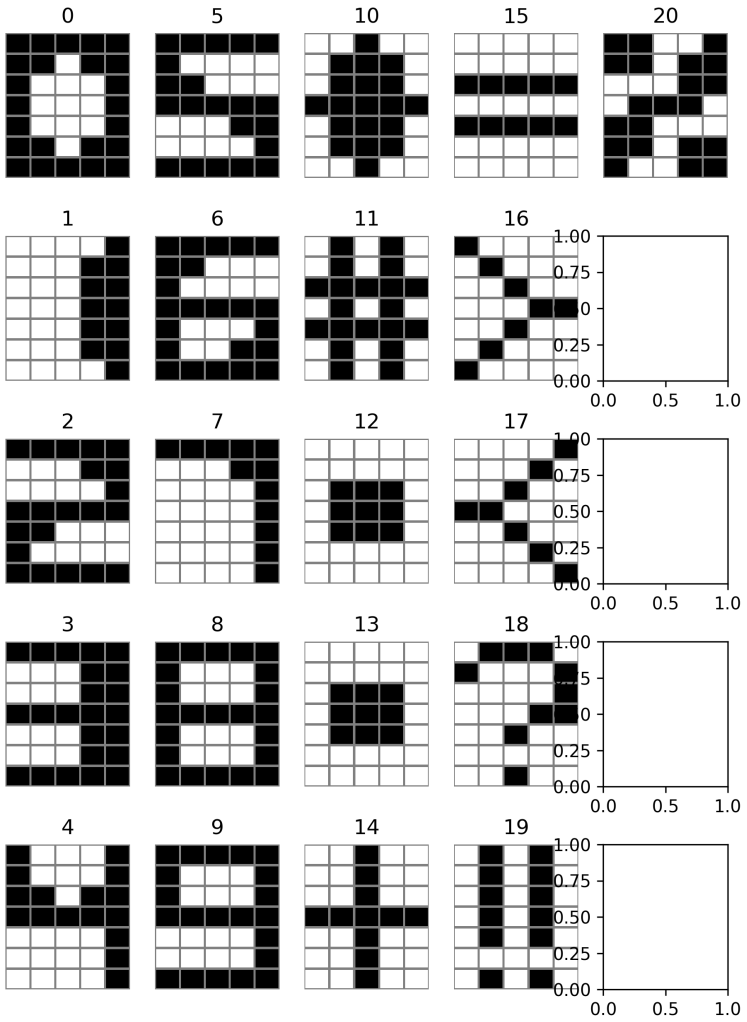

Figura 2: Gráfica de comportamiento.

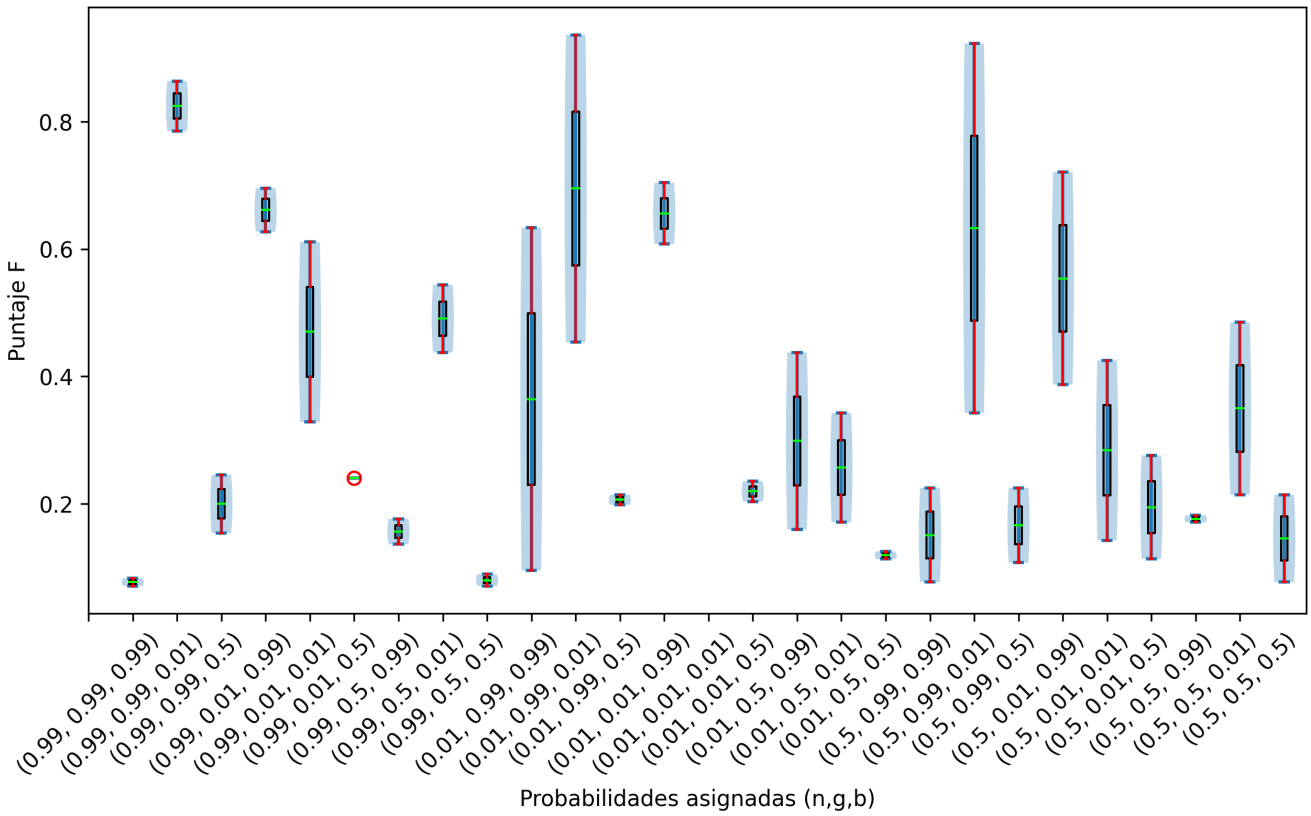

Figura 3: Grafica de comportamiento.

6. Reto 2

En el segundo reto, agrega ruido sal-y-pimienta en las entradas para una combinación ngb con la cual la red desempeña bien; este tipo de ruido se genera cambiando con una probabilidad pr los pixeles a blanco o negro (uniformemente al azar entre las dos opciones). Estudia el efecto de pr en el desempeño de la red (no importa si se hace esto con la red de la tarea base o la red extendida del primer reto).

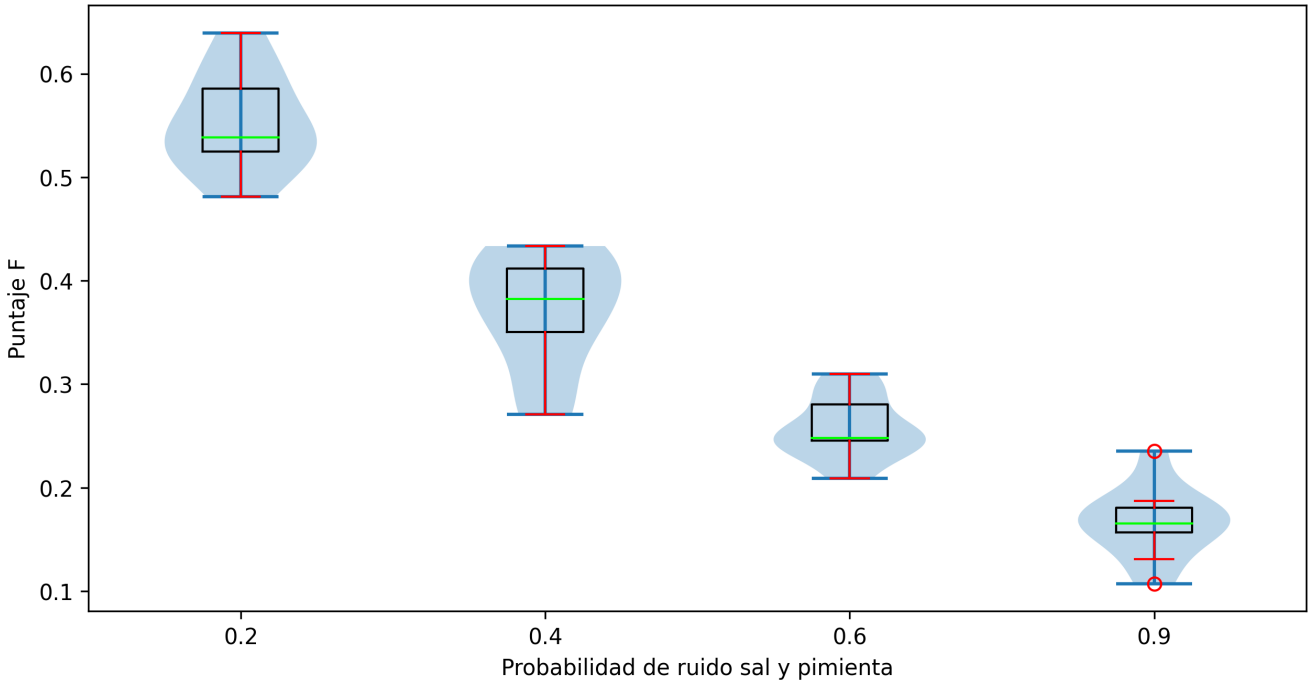

Figura 4: Grafica de comportamiento.

7. Conclusión

Se mostró con graficas de violín como va comportamiento al aumentar la probabilidad del ruido sal y pimienta no puede comprender que dato manejamos y eso hace que baje el porcentaje FC.

Referencias

[1] K. Gurney. An introduction to neural networks. 1997. URL `https://www.crcpress.com/An-Introduction-to-Neural-Networks/Gurney/p/book/9781857285031`.

[2] E. Schaeffer. Práctica 11: frentes de pareto. 2022. URL `https://satuelisa.github.io/simulation/p11.html`.