Práctica 12
Red neuronal

Edson Edgardo Samaniego Pantoja
Materia: Simulación computacional
Fecha 19 de mayo de 2021

1. Introducción

Esta última práctica consiste en la demostración básica de aprendizaje a máquina que consiste en reconocer dígitos de imágenes pequeñas en blanco y negro con una red neuronal. Un elemento básico de esta red es el perceptron que es un hiper plano que busca colocarse en la frontera que separa entradas verdaderas y las entradas falsas. Para la práctica es dado un código en el repositorio de Schaeffer [3] que es mejor explicado paso a paso en su pagina web [2], consiste en que el perceptron diga falso cuando la imagen es un cero y verdadero cuando la imagen contiene un uno, pero se extiende esta idea a los diez dígitos de cero a nueve. Lo que significa que se asocian los números a vectores de largo fijo de verdades y falsos, lo que se logra fácilmente con sus representaciones en números binarios.

Además de codificar las respuestas deseadas, se realiza el poder generar entradas que no sean todas idénticas para que haya algo de reto en obtener la respuesta correcta. Se crean imágenes de dígitos de una manera probabilística a partir de plantillas. Los píxeles que son negros en la plantilla serán puestas casi siempre, mientras los grises ocasionalmente, y los blancos solamente como ruido aleatorio de poca frecuencia.

2. Objetivo

Estudiar de manera sistemática el desempeño de la red neuronal en términos de su puntaje F (F-score) 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.

3. Código

Lo primero que se realiza en el código de Samaniego [1] es la forma en que se varian de manera dispersa en valores de `n,g,b` para poder observar qué factor es con el que mejor trabaja la red neuronal. Para eso se genera un dispersión en un gráfico de tres dimensiones para que cada punto represente el valor de `x,y,z` respectivamente a `n,g,b`. Se puede observar en figura 1.

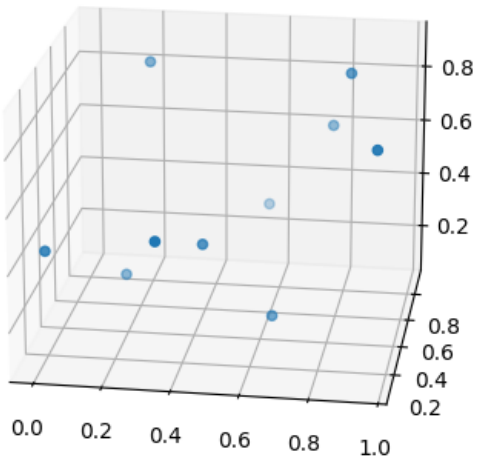

Figura 1: Gráfico de dispersión tres dimensiones para valores de `n,g,b`.

```
from pyDOE import lhs
var=3
muestras =10
x= lhs(var, muestras, criterion="corr")
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x[:,0], x[:,1], x[:,2])
plt.show()
```

Después se entra a un ciclo que varía cada punto de la dispersión o cada coordenada de tres dígitos y se extrae la posición de valor para `n,g,b` respectivamente. En la variable modelos donde se asignan los valores ya mencionados simplemente se remplazo por la variable que estará cambiando en cada ciclo.

```
for ciclos in range(0, muestras):

    posiciones= x[ciclos]
    posn=posiciones[0]
    posg=posiciones[1]
    posb=posiciones[2]
    modelos = pd.read_csv('digits.txt', sep=' ', header = None)
    modelos = modelos.replace({'n': posn, 'g': posg, 'b': posb})
    r, c = 5, 3
    dim = r * c
    tasa = 0.15
```

```
tranqui = 0.99
tope = 9
```

En cada ciclo una vez que termina el entrenamiento y la prueba, se extrae de la matriz de confusión generada el `F-score` para los datos. Obteniendo como final en la variable `precision` interpretada como porcentaje de qué tanto mejora el resultado reconociendo imágenes de números, con los valores dados en un inicio de `n,g,b`.

```
contadores2=np.delete(contadores,10,1)# elimina la columna 10
TP= np.diag(contadores2)
FP= np.sum(contadores2, axis=0)-TP
FN = np.sum(contadores2, axis=1) - TP
num_classes = 10
TN = []
for i in range(num_classes):
    temp = np.delete(contadores2, i, 0)
    temp = np.delete(temp, i, 1)
    TN.append(sum(sum(temp)))

precision= TP/(TP+FP)
```

4. Resultados

Los resultados pueden revisarse en la siguiente tabla 1 donde se puede ver para cada muestra que valores de negro gris y blanco se dieron y como afecto estos datos al reconocimiento de la red neuronal y viendo que mejora se ve según la condición.

Cuadro 1: Registro por muestra, valores de n,g,b y sus promedios resultantes en reconocimiento de dígitos.

Muestra	valor `n,g,b`	Promedio	Promedio (TP)
1	0.62, 0.45, 0.08	0.19	4.3
2	0.54, 0.93, 0.28	0.18	4.3
3	0.00, 0.17, 0.41	0.11	2.7
4	0.34, 0.06, 0.52	0.16	3.7
5	0.20, 0.73, 0.87	0.24	6.6
6	0.43, 0.34, 0.39	0.09	2.7
7	0.16, 0.57, 0.15	0.14	3.0
8	0.76, 0.83, 0.63	0.11	3.1
9	0.83, 0.62, 0.90	0.09	2.7
10	0.95, 0.29, 0.77	0.21	5.0

Los resultados de la tabla se pueden ver gráficamente en los diagramas caja bigote de la figura 2 realizados donde cada uno representa los datos obtenidos de la variable `precision` la cual tiene el porcentaje de efectividad en reconocimiento de cada número.

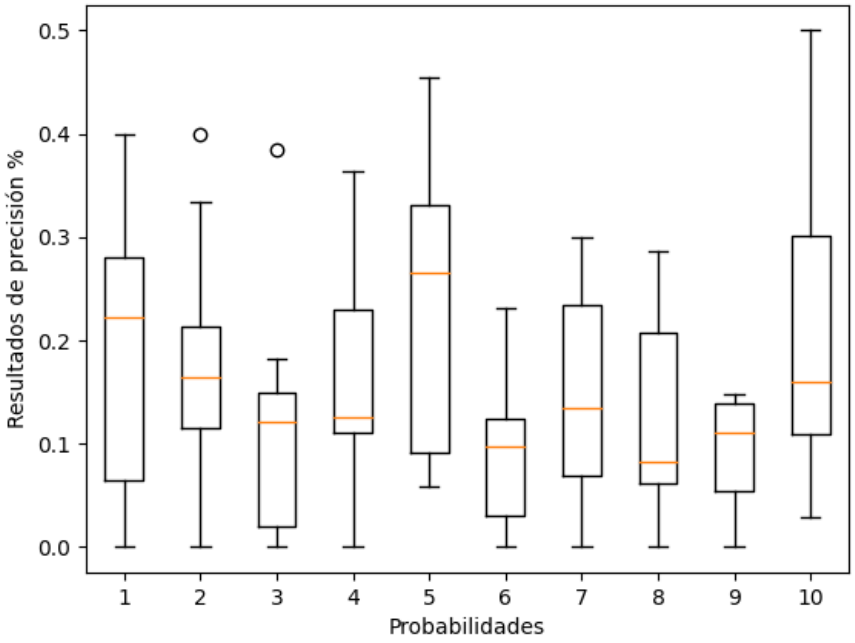

Figura 2: Caja bigote por cada muestra realizada y sus datos expresados en porcentaje de efectividad en el reconocimiento de números.

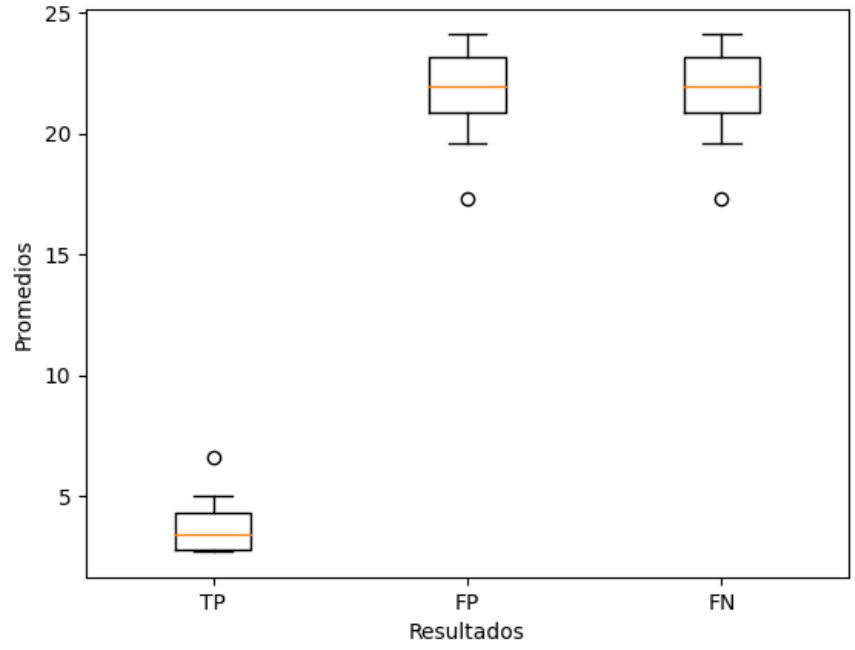

Figura 3: Caja bigote de los promedios que se obtienen de TP, FP, y FN.

5. Conclusión

En base a la tabla 1 y figura 2 se puede concluir en que la mejor combinación de negro, gris y blanco se da cuando la probabilidad de blanco y gris es mas grande o mas probable que la de negro ya que se vuelve mas flexible la manera de reconocimiento de los números dados. El porcentaje mas bajo se obtiene cuando la probabilidad de negro es mas alta que las otras dos pero gris y blanco se encuentran muy cercanos en sus probabilidades.

Referencias

[1] Samaniego E. Práctica 12, 2021. URL https://github.com/edson-samaniego/simulation-2021/tree/main/Practica-12.

[2] Schaeffer E. Red neuronal., 2021. URL https://github.com/satuelisa/Simulation/blob/master/NeuralNetwork/ann.py.

[3] Schaeffer E. *Práctica 12*, 2021. URL https://elisa.dyndns-web.com/teaching/comp/par/p12.html.