# Práctica 12 red neuronal

Denisse Leyva

Mayo 18, 2021

## 1. Introducción

En esta práctica se vera una demostración básica de aprendizaje de 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 $d$ el 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 $x * w$ es decir $\sum_{i=1}^{d} x_i = x_i w_i$ y si esta suma es positiva, la salida del perceptrón es verdad, en otro caso es falso. 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.

Cada error causa un movimiento en el perceptrón mientras los resultados correctos no le mueven. Las correcciones se hacen gradualmente más pequeñas por la reducción de la tasa de aprendizaje. Terminando el entrenamiento, la línea divisora ya no se mueve. En la fase de prueba, simplemente se clasifican las coordenadas recibidas como verdaderas ($x > y$) o falsas (en otro caso) [2].

## 2. Objetivo

Estudiar 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

El código base se sacó de Schaeffer [3]. El código completo se encuentra en el GitHub [1].

```
#Se crea el experimento factorial y se guarda en una matriz
fd = fullfact([3, 3, 3])
fd = fd/2
```

Código 1: Creación del experimento factorial.

Cuadro 1: Tabla experimento factorial.

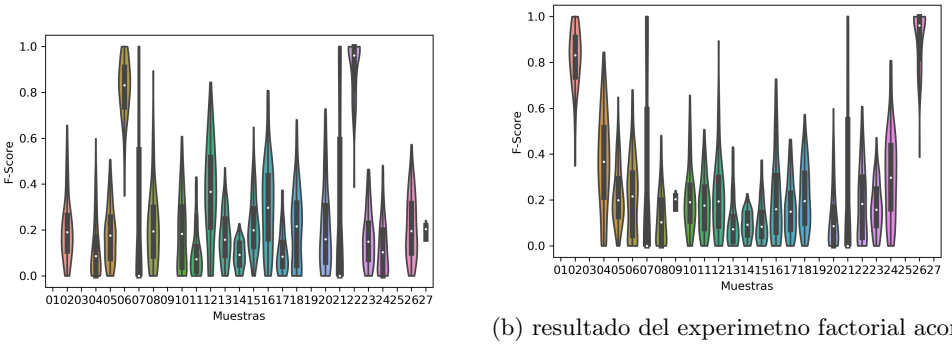
| Conteo | n | g | b |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0.5 | 0 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 0.5 | 0 |
| 5 | 0.5 | 0.5 | 0 |
| 6 | 1 | 0.5 | 0 |
| 7 | 0 | 1 | 0 |
| 8 | 0.5 | 1 | 0 |
| 9 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0.5 |
| 11 | 0.5 | 0 | 0.5 |
| 12 | 1 | 0 | 0.5 |
| 13 | 0 | 0.5 | 0.5 |
| 14 | 0.5 | 0.5 | 0.5 |
| 15 | 1 | 0.5 | 0.5 |
| 16 | 0 | 1 | 0.5 |
| 17 | 0.5 | 1 | 0.5 |
| 18 | 1 | 1 | 0.5 |
| 19 | 0 | 0 | 1 |
| 20 | 0.5 | 0 | 1 |
| 21 | 1 | 0 | 1 |
| 22 | 0 | 0.5 | 1 |
| 23 | 0.5 | 0.5 | 1 |
| 24 | 1 | 0.5 | 1 |
| 25 | 0 | 1 | 1 |
| 26 | 0.5 | 1 | 1 |
| 27 | 1 | 1 | 1 |

```
TP = np.diag(cm)
FP = np.sum(cm[:,:-1], axis=0) - TP
FN = np.sum(cm, axis=1) - TP
num_classes = k
TN = []
for i in range(num_classes):
    temp = np.delete(cm, i, 0)    # delete ith row
    temp = np.delete(temp, i, 1)  # delete ith column
    TN.append(sum(sum(temp)))
l = 300
for i in range(num_classes):
    print(TP[i] + FP[i] + FN[i] + TN[i] == l)
precision = TP/(TP+FP)
recall = TP/(TP+FN)
specificity = TN/(TN+FP)
F_score = 2/ ((1/precision)+(1/recall))
F_score2 = TP / (TP+((FP+FN)/2))
```

Código 2: Obtención del F score.

# 4. Resultados

Como se puede observar existen cuatro combinaciones donde el F score obtuvo los mejores resultados en la combinación "1 0 0", "1 1 0", "0 0 1", "0 1 1", así mismo entre las peores combinaciones que podemos encontrar están la "0.5 0.5 0.5", "1 1 1", "0.5 0 0.5", "0.5 1 0.5". Podemos concluir que para obtener mejores resultados una opción sería mantener a negro y gris altos y otra opción sería mantener blancos altos y negro y gris lo más bajo posible.




(a) resultado del experimento factorial.

(b) resultado del experimetno factorial acomodando $n$ de 1 a 0.

Figura 1: Gráficas de resultados obtenidos con el experimento factorial.

# 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 x 7 de lo original de 3 x 5 (modificando las plantillas de los dígitos acorde a este cambio) [2].

Para este reto se agregó un perceptrón más por la cantidad de dígitos disponibles, y se puede observar en la gráfica de resultados que al momento de tener más dígitos el F score no llega totalmente a 1 como la tarea base. Pero la conclusión es la misma para tener mejores resultados las opciones serian iguales a la tarea base.

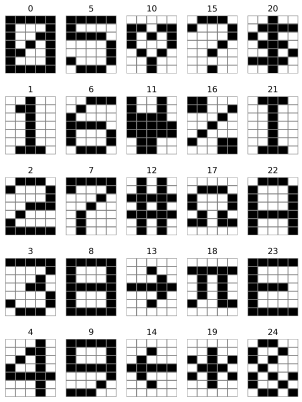
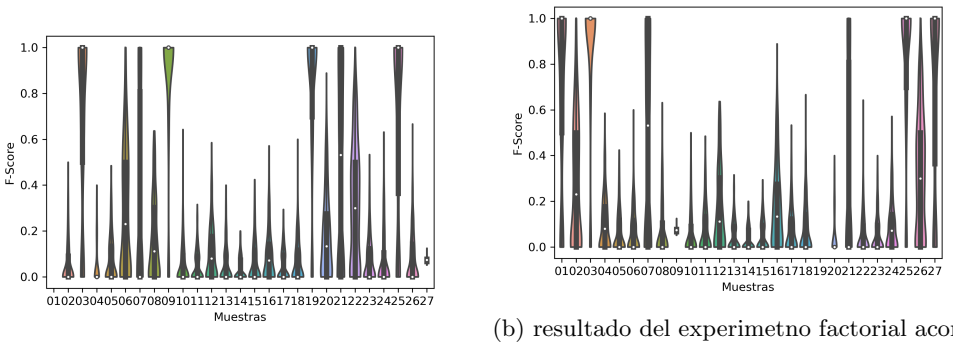


Figura 2: Símbolos ASCII usados para el experimento factorial.




(a) resultado del experimento factorial.

(b) resultado del experimetno factorial acomodando $n$ de 1 a 0.

Figura 3: Gráficas de resultados obtenidos con el experimento factorial.

# 6. Reto 2

En este segundo reto se 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 $p_r$ los pixeles a blanco o negro (uniformemente al azar entre las dos opciones). Estudia el efecto de $p_r$ 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) [2].

Para este reto se escogió la combinación de "1 1 0"que es de las mejores combinaciones en la tarea base, a esta combinación se le agrego ruido sal y pimienta que va con un porcentaje de probabilidad de 10 % del 0 al 100.

```
for t in range(5000): # entrenamiento
    d = randint(0, tope)
    pixeles = abs(1 * (np.random.rand(dim) < modelos.iloc[d]) - (np.random.uniform(size = dim)<pr))
    correcto = '{0:04b}'.format(d)
    for i in range(n):
        w = neuronas[i, :]
        deseada = int(correcto[i]) # 0 o 1
        resultado = sum(w * pixeles) >= 0
        if deseada != resultado:
            ajuste = tasa * (1 * deseada - 1 * resultado)
            tasa = tranqui * tasa
            neuronas[i, :] = w + ajuste * pixeles
```

Código 3: Para agregar ruido sal y pimienta.

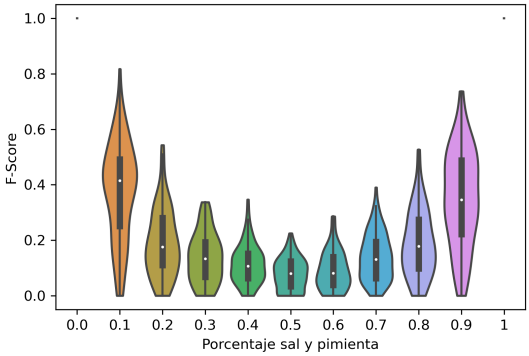



Figura 4: Gráfica mostrando el porcentaje del ruido.

# Referencias

[1] *Denisse Leyva Repositorio*. URL: `https://github.com/Denisse251/Simulation/tree/main/Tarea.12`.

[2] *Elisa Schaeffer Práctica 12*. URL: `https://elisa.dyndns-web.com/teaching/comp/par/p12.html`.

[3] *Elisa Schaeffer Repositorio*. URL: `https://github.com/satuelisa/Simulation`.