

Universidad del Valle de Guatemala  
Departamento de Matemática  
Licenciatura en Matemática Aplicada

**Estudiante:** Rudik Roberto Rompich  
**Correo:** rom19857@uvg.edu.gt  
**Carné:** 19857

CC3066 - Data Science I - Catedrático: Luis Furlan  
26 de agosto de 2021

---

## Laboratorio 4: Redes Neuronales Básicas

**Instrucciones:** en clase vimos un modelo simple para resolver regresiones lineales mediante redes neuronales. Utilizado el código desarrollado (o si lo desea uno propio), responda a las siguientes preguntas:

Para realizar este laboratorio se hicieron algunas modificaciones al código proporcionado por el catedrático:

1. Se utilizó *seaborn* en lugar de *plotly*, debido a que las gráficas ya no se generaban cuando las observaciones eran demasiado grandes.
2. Para determinar los tiempos de ejecución, **no se corrieron las gráficas** (ya que es la parte del código que consume más tiempo) y lo importante son los 3 datos que se obtuvieron.
3. Se modificaron las iteraciones y se colocaron 6000 con un  $\eta = 0.1$  (eta); ya que con ese número en todos los casos daba los números esperados (y no era necesario correr varias veces la celda anterior, como se indicaba en las instrucciones).
4. Para medir el tiempo de ejecución se utilizó la librería de *time*.

**Problema 1.** *Cambie el número de observaciones a 100,000. Explique que es lo que ocurre en términos de:*

1. *El tiempo de ejecución para resolver el problema.*

**Solución.** El tiempo de ejecución en segundos es un poco más prolongado; pero no excesivamente, alrededor de 9 segundos.

```
[25] end = time.time()
      print(end - start)

9.019009828567505
```

□

2. El resultado final versus lo encontrado en clase: es igual, o diferente...¿por qué?

**Solución.** Es prácticamente igual, mismos pesos y sesgo.

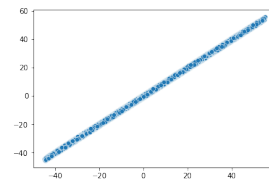
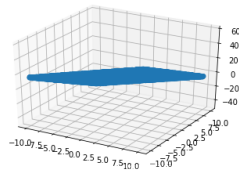
```
[23] print(weights, biases)

[[ 1.99991049]
 [-2.99980964]] [4.9863994]
```

□

3. Las gráficas para representar los datos/resultados.

**Solución.** Las gráficas con 100,000 observaciones no varían en prácticamente nada la forma con las 1000 iteraciones iniciales.



□

**Problema 2.** Cambie el número de observaciones a 1,000,000. Explique que es lo que ocurre en términos de:

1. El tiempo de ejecución para resolver el problema.

**Solución.** El tiempo de ejecución en segundos es relativamente grande, rondando en alrededor de los 2 minutos de tiempo de ejecución (esto influyó por el número de iteraciones).

```
[13] end = time.time()
      print(end - start)

104.1334900856018
```

□

2. El resultado final versus lo encontrado en clase: es igual, o diferente...¿por qué?

**Solución.** El resultado es prácticamente igual, mismos pesos y sesgo.

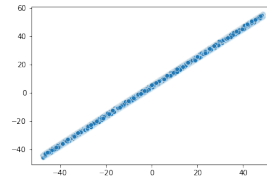
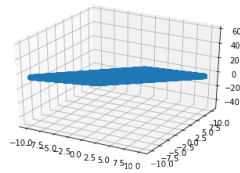
```
[35] print(weights, biases)

[[ 2.0000596 ]
 [-3.00005813]] [4.98811275]
```

□

3. Las gráficas para representar los datos/resultados.

**Solución.** Las gráficas con 1,000,000 observaciones no varía en absolutamente nada comparado a las 100,000 y 1000 observaciones. Quizás simplemente son más robustas.



9

**Problema 3.** “Juegue” un poco con el valor de la tasa de aprendizaje, por ejemplo 0.0001, 0.001, 0.1, 1. Para cada uno de estos indique:

1. ¿Qué ocurre con el tiempo de ejecución?

**Solución.** El tiempo de ejecución va aumentando cuando el  $\eta$  va aumentando, por lo tanto es directamente proporcional.  $\square$

2. ¿Qué ocurre con la minimización de la pérdida?

**Solución.** En los primeros dos casos, la pérdida va disminuyendo. Sin embargo, para los últimos dos casos la pérdida se vuelve increíblemente grande.  $\square$

### 3. ¿Qué ocurre con los pesos y los sesgos?

**Solución.** En todos los casos los pesos son iguales; únicamente el sesgo varía drásticamente.  $\square$

#### 4. ¿Qué ocurre con las iteraciones?

**Solución.** Se utilizaron 6000 iteraciones en todos los casos, ya que en la pregunta 1 y 2 se determinó que era el número ideal para encontrar los pesos y sesgo correcto.

5. ¿El problema queda resuelto o no?

**Solución.** Sí, pero únicamente cuando el  $\eta$  es pequeño.  $\square$

6. ¿Cuál es la apariencia de la última gráfica? ¿Se cumple con la condición de que sea de 45 grados?

**Solución.** No, la última gráfica explota en los últimos dos casos; ya que los números son demasiado grandes.  $\square$

**Problema 4.** *Cambie la función de pérdida “L2-norm” a la misma pero sin dividir por 2. Explique lo que ocurre en términos de:*

1. El tiempo que se tarda el algoritmo en terminar, comparado a lo que vimos en clase.

**Solución.** El tiempo incrementa, aproximadamente el doble. ☐

2. Si la pérdida se minimiza igual que lo que vimos en clase.

**Solución.** En efecto, la pérdida se minimiza. ☐

3. Si los pesos y sesgos son parecidos a los vistos en clase.

**Solución.** Sí, los pesos son los mismos; el sesgo varía un poco. ☐

4. Si el problema se resuelve como ocurrió en clase.

**Solución.** Sí, el problema también se resolvió quitándole el 2. ☐

5. Si se obtiene un mejor resultado al hacer más iteraciones.

**Solución.** Es una pregunta más o menos inconclusa; ya que al utilizar más iteraciones se obtiene el sesgo correcto un poquito más antes. ☐

**Problema 5.** Cambie la función de pérdida de la “L2-norm” a la “L1-norm”. Explique lo que ocurre en términos de:

1. El tiempo que se tarda el algoritmo en terminar, comparado a lo que vimos en clase

**Solución.** Levemente más alto, alrededor de 1 segundo más tardado. ☐

2. Si la pérdida se minimiza igual que lo que vimos en clase

**Solución.** No, ahora va de un número negativo hasta llevar a acercarse a 0. ☐

3. Si los pesos y sesgos son parecidos a los vistos en clase

**Solución.** Sí, son similares nuevamente los pesos; únicamente el sesgo varía. ☐

4. Si el problema se resuelve como ocurrió en clase

**Solución.** Sí, el problema igualmente se resolvió. ☐

5. Si se obtiene un mejor resultado al hacer más iteraciones.

**Solución.** Así es, a más iteraciones; mejor resultado. El número de iteraciones propuesto es de 6000. ☐


6. ¿Tendrá una de estas más limitaciones que la otra?

**Solución.** Es probable que su limitante sea el tiempo. ☐

**Problema 6.** Cree una función  $f(x, z) = 13 * x_s + 7 * z_s - 12$ .

1. *¿Funciona el algoritmo de la misma forma?*

**Solución.** Se modificó el código con la siguiente celda:

```
 ruido = np.random.uniform(-1, 1, (observaciones,1))  
targets = 13 * x1 + 7 * x2 - 12 + ruido  
  
# Veamos las dimensiones. Deben ser n x m, donde m es el número de variables de salida.  
print (targets.shape)  
  
(1000, 1)
```

En conclusión, el algoritmo funciona de la misma manera, ya que solo se está modificando el *target*; pero todo lo demás funciona exactamente igual. □