# Convolutional Neural Networks – Lab 3

RUBEN MARTINEZ GONZALEZ

February 14, 2024

## 1 introducción

En este informe, se presenta el trabajo realizado en el laboratorio 3 de Redes Neuronales Convolucionales. Se implementaron varios bloques de código relacionados con el uso de la regresión lineal y el algoritmo de descenso de gradiente. Se ejemplificó el uso de la regresión lineal para predecir la altura de varios niños basándose en sus edades.
La solución se implementó en Python utilizando la biblioteca NumPy para el procesamiento de matrices y la biblioteca Matplotlib para la visualización de gráficos. La implementación está desplegada en un notebook de Google Colab, el cual se puede acceder a través del siguiente enlace: `Colab`

## 2 Bloques de código implementados

### 2.1 Linear Regression using Gradient Descent

En este bloque de código se implementa la regresión lineal utilizando el algoritmo de descenso de gradiente. El objetivo de la implementación es poder predecir la altura de varios niños basándose en sus edades. Se utilizó una tasa de aprendizaje de $\alpha = 0.07$ y se realizaron 1000 iteraciones del algoritmo de descenso de gradiente. Los valores de $x$ y $y$ se cargaron desde los archivos 'ex2x.dat' y 'ex2y.dat' respectivamente.

```
import numpy as np
from random import random

# Load the data
X_train = np.loadtxt('ex2x.dat')
y_train = np.loadtxt('ex2y.dat')

# Initialize variables
m = len(X_train)      # Number of training examples
alpha = 0.07          # Learning rate
theta = [random() - 0.5 for _ in range(2)] # Initial random parameters

# Gradient descent iterations
for i in range(1000):
  # Predict heights based on current parameters
  y_pred = theta[0] + theta[1] * X_train # h (x)= 0 + 1 * x

  # Calculate the error
  error = y_pred - y_train

  # Update parameters using gradient descent algorithm
  theta[0] -= alpha * (1/m) * np.sum(error)            #  0 :=  0  -    *(1/m)
     [h (xi) - yi]
  theta[1] -= alpha * (1/m) * np.sum(error * X_train)  #  1 :=  1  -    *(1/m)
     [h (xi) - yi]*xi

```

```
# Print the learned parameters
print('Theta0:', theta[0])
print('Theta1:', theta[1])

```

Listing 1: Regresión lineal utilizando descenso de gradiente

La implementación realizada se describe a continuación:

- Se cargan los datos de entrenamiento desde los archivos 'ex2x.dat' y 'ex2y.dat'.
- Se inicializan las variables 'm' en la cantidad de ejemplos de entrenamiento en el archivo 'ex2x.dat' (cantidad de niños con datos de altura)
- y '$\alpha$' (tasa de aprendizaje) en 0.07.
- Se inicializan los parámetros $\theta$ con valores aleatorios entre -0.5 y 0.5.
- Se realizan 1000 iteraciones del algoritmo de descenso de gradiente y en cada una de ellas:
  - Se realizan las predicciones de las alturas basadas en los parámetros actuales, para la predicción se utiliza la fórmula $h\theta$(x)= $\theta$0 + $\theta$1 * x
  - Se calcula el error por diferencia entre las predicciones y las alturas reales.
  - Se actualizan los parámetros utilizando el algoritmo de descenso de gradiente donde:
    - $\theta_0$ := $\theta_0$ - $\alpha$ * (1/m) * $\sum$ [h$\theta$($x_i$) - $y_i$]
    - $\theta_1$ := $\theta_1$ - $\alpha$ * (1/m) * $\sum$ [h$\theta$($x_i$) - $y_i$] * $x_i$
  - Se imprimen los parámetros aprendidos.

  Al ejecutar el código se obtuvieron los siguientes resultados:

- Theta0: 0.749921461773668
- Theta1: 0.06392502646496022

## 2.2 Plot your training data

En este bloque de código se grafican los datos de entrenamiento, donde se muestra la relación entre la edad y la altura de los niños.

```
import matplotlib.pyplot as plt

plt.scatter(X_train, y_train, marker='x', c='r')
plt.title('Training Data')
plt.xlabel('Age in years')
plt.ylabel('Height in meters')
plt.show()

```

Listing 2: Gráfica de los datos de entrenamiento

Al ejecutar el código se obtuvo la siguiente gráfica:


Figure 1: Gráfica de los datos de entrenamiento

## 2.3 Initialize the parameters to $\theta = 0$

En este bloque de código se inicializan los parámetros $\theta$ a 0 y se ejecuta una iteración del algoritmo de descenso de gradiente.

```
# Initialize variables
m = len(X_train)      # Number of training examples
alpha = 0.07          # Learning rate
theta = np.zeros(2)   # Initial parameters

# Gradient descent iterations
for i in range(1):
# Predict heights based on current parameters
y_pred = theta[0] + theta[1] * X_train # h (x)= 0 + 1 * x

# Calculate the error
error = y_pred - y_train

```

```
# Update parameters using gradient descent algorithm
theta[0] -= alpha * (1/m) * np.sum(error)            #  0 :=  0  -    *(1/m)     [
    h  (xi) - yi]
theta[1] -= alpha * (1/m) * np.sum(error * X_train)  #  1 :=  1  -    *(1/m)     [
    h  (xi) - yi]*xi

# Print the parameters after one iteration
print('Theta0 after one iteration:', theta[0])
print('Theta1 after one iteration:', theta[1])
```

Listing 3: Inicialización de los parámetros a $\theta = 0$

Al ejecutar el código se obtuvieron los siguientes resultados:

- Theta0 after one iteration: 0.07452802382200001
- Theta1 after one iteration: 0.38002167250780633

## 2.4 Continue running gradient descent for more iterations

En este bloque de código se ejecutan 1000 iteraciones del algoritmo de descenso de gradiente.

```
# Continue gradient descent for more iterations
for i in range(1000):
# Predict heights based on current parameters
y_pred = theta[0] + theta[1] * X_train # h  (x)=  0  +  1  * x

# Calculate the error
error = y_pred - y_train

# Update parameters using gradient descent algorithm
theta[0] -= alpha * (1/m) * np.sum(error)            #  0 :=  0  -    *(1/m)     [
    h  (xi) - yi]
theta[1] -= alpha * (1/m) * np.sum(error * X_train)  #  1 :=  1  -    *(1/m)     [
    h  (xi) - yi]*xi

# Print the final parameter values
print('Final Theta0:', theta[0])
print('Final Theta1:', theta[1])
```

Listing 4: Ejecución de 1000 iteraciones del algoritmo de descenso de gradiente

Al ejecutar el código se obtuvieron los siguientes resultados:

- Final Theta0: 0.749921461773668
- Final Theta1: 0.06392502646496022

## 2.5 Plot the straight line fit from your algorithm

En este bloque de código se grafica la línea recta ajustada por los parámetros aprendidos por el algoritmo de descenso de gradiente. Se grafican los datos de entrenamiento y la línea recta ajustada por los parámetros aprendidos por el algoritmo de descenso de gradiente. Se muestra la relación entre la edad y la altura de los niños.

```
# Plot the training data and the fitted line
plt.scatter(X_train, y_train, marker='x', c='r', label='Training Data')
plt.plot(X_train, theta[0] + theta[1] * X_train, color='b', label='Fitted Line')
plt.title('Training Data with Fitted Line')
plt.xlabel('Age in years')
plt.ylabel('Height in meters')
```

```
plt.legend()
plt.show()
```

Listing 5: Gráfica de la línea recta ajustada por el algoritmo de descenso de gradiente

Al ejecutar el código se obtuvo la siguiente gráfica:

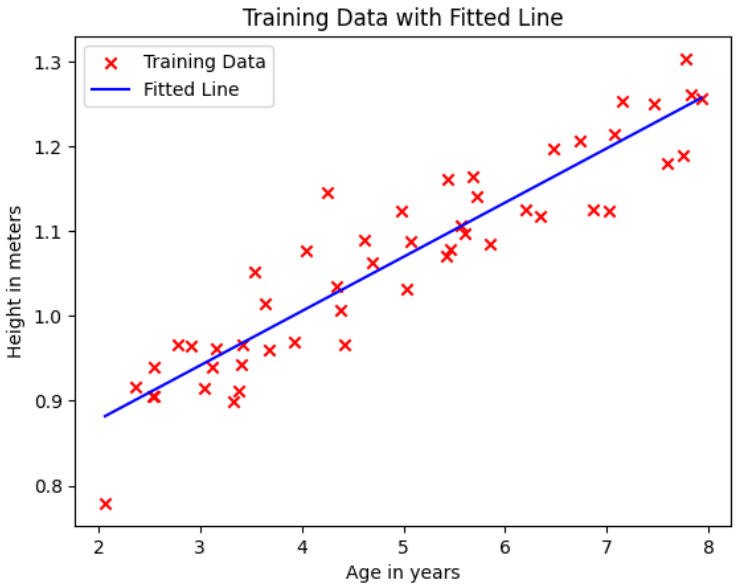


Figure 2: Gráfica de la línea recta ajustada por el algoritmo de descenso de gradiente

## 2.6 Use your trained model to predict height

En este bloque de código se utilizan los parámetros aprendidos por el algoritmo de descenso de gradiente para predecir la altura de dos niños de 3.5 y 7 años.

```
# Predict heights for boys of age 3.5 and 7
age_3_5_height = theta[0] + theta[1] * 3.5
age_7_height = theta[0] + theta[1] * 7

# Print the predicted heights
print('Predicted height for a boy of age 3.5:', round(age_3_5_height,2))
print('Predicted height for a boy of age 7:', round(age_7_height,2))
```

Listing 6: Predicción de la altura de dos niños de 3.5 y 7 años

Al ejecutar el código se obtuvieron los siguientes resultados:

- Predicted height for a boy of age 3.5: 0.97
- Predicted height for a boy of age 7: 1.2

# 3 Conclusiones

En este informe se presentó el trabajo realizado en el laboratorio 3 de Redes Neuronales Convolucionales. Se implementaron varios bloques de código relacionados con el uso de la regresión lineal y el algoritmo de descenso de gradiente. se ejemplificó el uso de la regresión lineal para predecir la altura de varios niños basándose en sus edades. Se utilizó una tasa de aprendizaje de $\alpha = 0.07$ y se realizaron 1000 iteraciones del algoritmo de descenso de gradiente. Se graficaron los datos de entrenamiento y la línea recta ajustada por los parámetros aprendidos por el algoritmo de descenso de gradiente. Se utilizó el modelo entrenado para predecir la altura de dos niños de 3.5 y 7 años.