Convolutional Neural Networks – Lab 5

RUBEN MARTINEZ GONZALEZ

March 6, 2024

1 introducción

En este informe, se presenta el trabajo realizado en el laboratorio 5 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 TensorFlow para el aprendizaje, NumPy para el procesamiento de matrices y 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 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

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

            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 1: Gráfica de los datos de entrenamiento

Al ejecutar el código se obtuvo la siguiente gráfica:

Figure 1: Gráfica de los datos de entrenamiento

2.2 Linear Regression using Gradient Descent (TensorFlow)

En este bloque de código se implementa la regresión lineal utilizando el algoritmo de descenso de gradiente apoyado en la biblioteca TensorFlow. El objetivo de la implementación es poder predecir la altura de varios niños basándose en sus edades. Los valores de x y y se cargaron desde los archivos 'ex2x.dat' y 'ex2y.dat' respectivamente.
La implementación realizada se describe a continuación:

- Se normalizaron los datos de entrada y salida.
- Se inicializaron las variables m (número de ejemplos de entrenamiento), α (tasa de aprendizaje = 0.1) y θ (parámetros iniciales aleatorios con distribución normal).
- Se definió la función de hipótesis $h_\theta(x) = \theta_0 + \theta_1 * x$.
- Se definió la función de costo (error cuadrático medio) $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$.
- Se definió el optimizador (descenso de gradiente estocástico).
- Se ejecutó una iteración del algoritmo de descenso de gradiente.

Al ejecutar el código se obtuvieron los siguientes resultados:

- Cost after 1 iterations: 3.884464
- Theta0 after one iteration: tf.Tensor([-0.09925186], shape=(1,), dtype=float32)
- Theta1 after one iteration: tf.Tensor([-0.34086537], shape=(1,), dtype=float32)

```
# Normalize the input data
X_mean = np.mean(X_train)
X_std = np.std(X_train)
X_train = (X_train - X_mean) / X_std

# Normalize the output data
y_mean = np.mean(y_train)
y_std = np.std(y_train)
y_train = (y_train - y_mean) / y_std

# Initialize variables
m = len(X_train)      # Number of training examples
alpha = 0.1           # Learning rate
theta = tf.Variable(tf.random.normal([2, 1]), dtype=tf.float32) # Initial random
    parameters

# Define the model
def hypothesis(X):
    return theta[0] + theta[1] * X

# Define the cost function (mean squared error)
def cost_function(X, y):
    return tf.reduce_mean(tf.square(hypothesis(X) - y))

# Define the optimizer
optimizer = tf.keras.optimizers.SGD(learning_rate=alpha)

with tf.GradientTape() as tape:
    cost = cost_function(X_train, y_train)
    gradients = tape.gradient(cost, [theta])
    optimizer.apply_gradients(zip(gradients, [theta]))
    print("Cost after", 1, "iterations:", cost.numpy())
```

```
32      print('Theta0 after one iteration:', theta[0])
33      print('Theta1 after one iteration:', theta[1])
34
```

Listing 2: Regresión lineal utilizando descenso de gradiente

2.3 Continue running gradient descent for more iterations

En este bloque de código se ejecutan 200 iteraciones del algoritmo de descenso de gradiente.

```
# Perform gradient descent
for i in range(300):
    # Perform one step of gradient descent
    with tf.GradientTape() as tape:
        cost = cost_function(X_train, y_train)
    gradients = tape.gradient(cost, [theta])
    optimizer.apply_gradients(zip(gradients, [theta]))

    # Print cost every 50 iterations
    if i % 50 == 0:
        print("Cost after", i, "iterations:", cost.numpy())

# Get the learned parameters
final_theta = theta.numpy()
print('Final Theta:', final_theta.ravel())

```

Listing 3: Ejecución de 200 iteraciones del algoritmo de descenso de gradiente

Al ejecutar el código se obtuvieron los siguientes resultados:

- Cost after 0 iterations: 1.7575389
- Cost after 50 iterations: 0.14193676
- Cost after 100 iterations: 0.14193676
- Cost after 150 iterations: 0.14193676
- Final Theta: [1.3125225e-08 9.2631692e-01]

2.4 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 junto a los datos de entrenamiento. 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, final_theta[0] + final_theta[1] * X_train, color='b', label='
    Fitted Line')
plt.title('Training Data with Fitted Line')
plt.xlabel('Age in years (normalized)')
plt.ylabel('Height in meters (normalized)')
plt.legend()
plt.show()

```

Listing 4: Gráfica de la línea recta ajustada por el algoritmo de descenso de gradiente

Al ejecutar el código se obtuvo la siguiente gráfica:

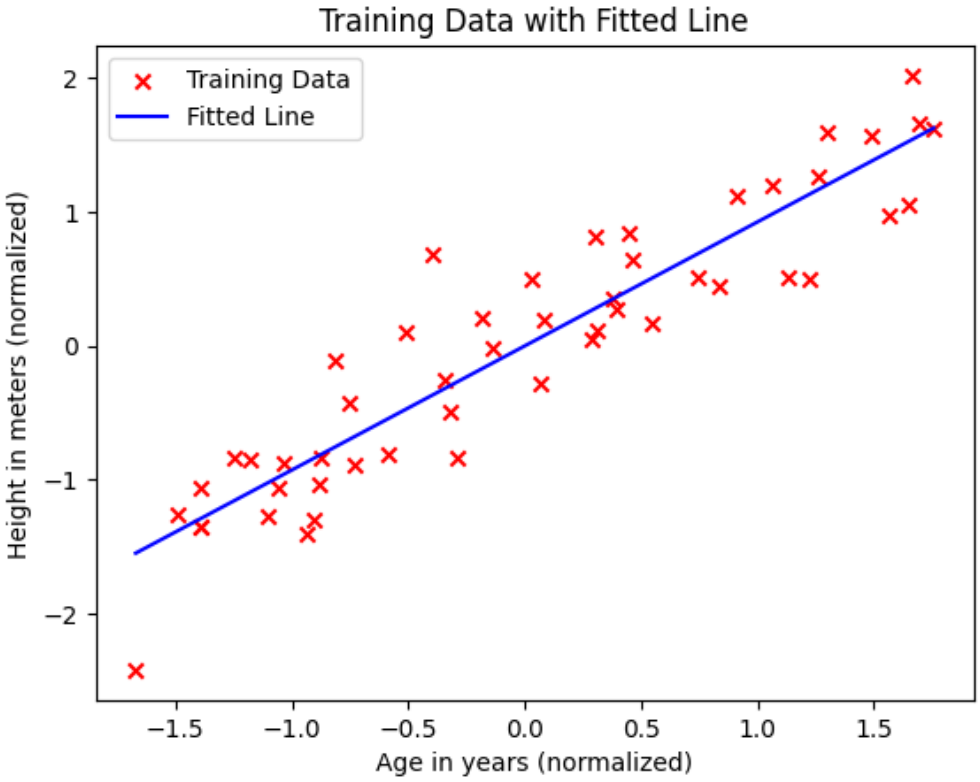

Figure 2: Gráfica de la línea recta ajustada por el algoritmo de descenso de gradiente

2.5 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 5: 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.