**ЛАБОРАТОРНА РОБОТА № 8**

РЕСУРСИ KERAS. TENSORFLOW. НАВЧАННЯ ЛІНІЙНОЇ РЕГРЕСІЇ.

***Мета:*** Дослідження ресурсу Keras і TensorFlow. Застосування TensorFlow.

**Хід роботи:**

Посилання на GitHub:

**Завдання 1**

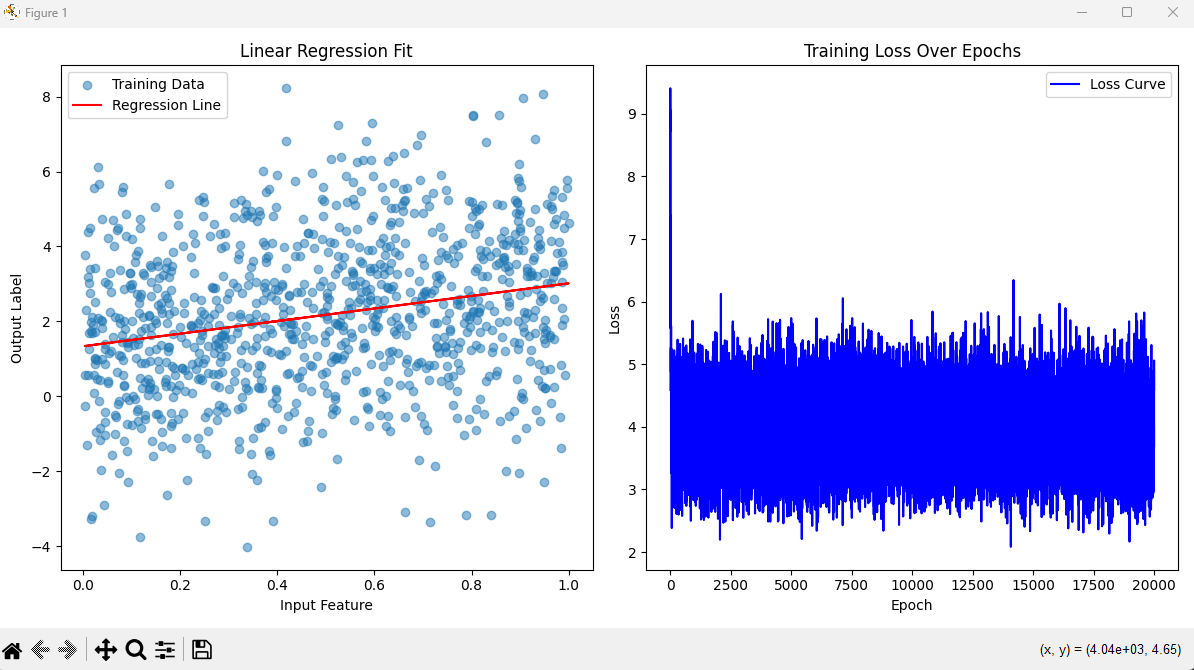
Кластеризація даних за допомогою методу k-середніх

**Програмний код:**

import numpy as np  
import tensorflow as tf  
import matplotlib  
import matplotlib.pyplot as plt  
  
matplotlib.use('TkAgg')  
  
np.random.seed(42)  
  
input\_features = np.random.rand(1000, 1).astype(np.float32)  
noise\_component = np.random.normal(0, 2, size=(1000, 1)).astype(np.float32)  
output\_labels = 2 \* input\_features + 1 + noise\_component  
  
slope = tf.Variable(tf.random.normal([1]), name='slope')  
intercept = tf.Variable(tf.zeros([1]), name='intercept')  
  
learning\_rate = 0.01  
num\_epochs = 20000  
batch\_size = 100  
  
sgd\_optimizer = tf.optimizers.SGD(learning\_rate)  
  
def calculate\_loss(actual, predicted):  
 return tf.reduce\_mean(tf.square(actual - predicted))  
  
loss\_history = []  
  
for epoch in range(num\_epochs):  
 random\_indices = np.random.choice(len(input\_features), batch\_size)  
 x\_batch = input\_features[random\_indices]  
 y\_batch = output\_labels[random\_indices]

with tf.GradientTape() as tape:  
 predictions = slope \* x\_batch + intercept  
 current\_loss = calculate\_loss(y\_batch, predictions)  
 gradients = tape.gradient(current\_loss, [slope, intercept])  
 sgd\_optimizer.apply\_gradients(zip(gradients, [slope, intercept]))  
  
 loss\_history.append(current\_loss.numpy())  
  
 if (epoch + 1) % 1000 == 0:  
 print(f"Epoch {epoch + 1}: Loss={current\_loss.numpy():.4f}, "  
 f"Slope={slope.numpy()[0]:.4f}, Intercept={intercept.numpy()[0]:.4f}")  
  
print(f"Final model parameters: Slope={slope.numpy()[0]:.4f}, Intercept={intercept.numpy()[0]:.4f}")  
  
plt.figure(figsize=(12, 6))  
  
plt.subplot(1, 2, 1)  
plt.scatter(input\_features, output\_labels, label='Training Data', alpha=0.5)  
plt.plot(input\_features, slope.numpy() \* input\_features + intercept.numpy(),  
 color='red', label='Regression Line')  
plt.title('Linear Regression Fit')  
plt.xlabel('Input Feature')  
plt.ylabel('Output Label')  
plt.legend()  
  
plt.subplot(1, 2, 2)  
plt.plot(range(num\_epochs), loss\_history, color='blue', label='Loss Curve')  
plt.title('Training Loss Over Epochs')  
plt.xlabel('Epoch')  
plt.ylabel('Loss')  
plt.legend()  
  
plt.tight\_layout()  
plt.show()

Результат виконання:





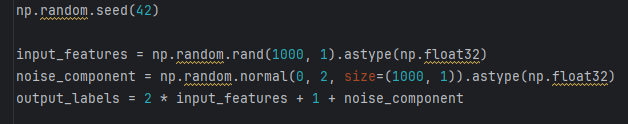
**Дослідження розрахункового алгоритму**

У даній роботі досліджується алгоритм лінійної регресії з використанням стохастичного градієнтного спуску (SGD) для визначення параметрів лінійної моделі. Алгоритм реалізований з використанням бібліотеки TensorFlow, а дані генеруються штучно для імітації задачі регресії.

1. **Створення початкових даних**

Алгоритм починається з генерації випадкових даних, які слідують лінійній залежності y=2x+1+шумy = 2x + 1 + \text{шум}. Генерація даних проводиться у два кроки: створення випадкових значень для xx та додавання шуму, щоб моделювати реальні дані.

**Код генерації даних:**

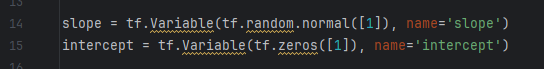
****

Це дозволяє згенерувати 1000 точок, де кожна точка xx має відповідний вихід yy з доданим випадковим шумом. Шум забезпечує, що дані не є ідеальними, як це буває в реальних задачах.

1. **Ініціалізація моделі**

Модель лінійної регресії представлена у вигляді параметрів: коефіцієнт нахилу (slope) та вільний член (intercept). Параметри ініціалізуються випадковими значеннями.

**Код ініціалізації:**

****

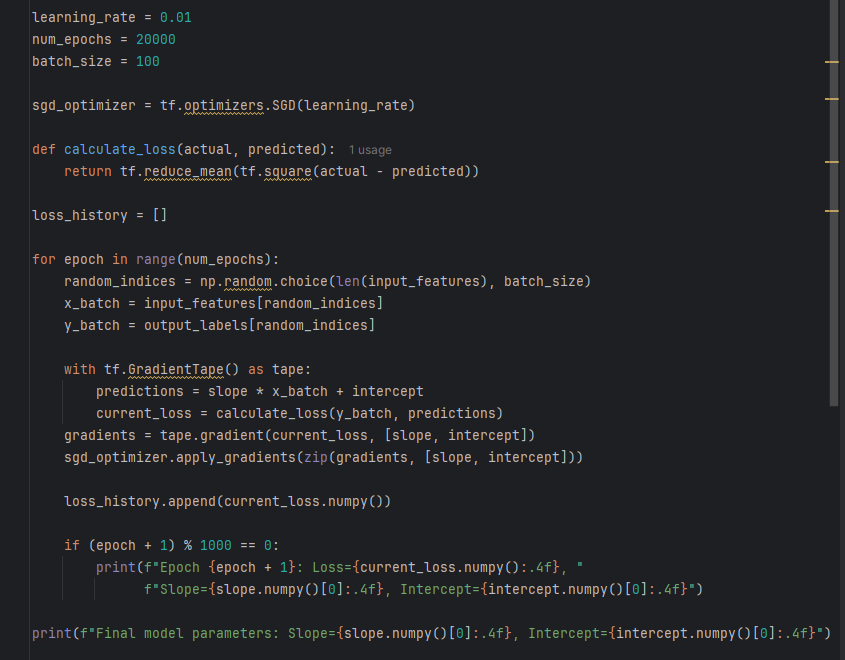
• slope – параметр, що відповідає за нахил лінії регресії.

• intercept – зсув лінії відносно осі y.

3**. Процес навчання (Стохастичний градієнтний спуск)**

Для навчання моделі використовується стояхастичний градієнтний спуск (SGD), який оновлює параметри на основі випадково обраної підмножини даних (batch). Оновлення параметрів відбувається шляхом мінімізації функції втрат, що обчислює середньоквадратичну помилку (MSE)

**Код процесу навчання:**

****

На кожному кроці:

1. Випадково обирається міні-батч розміром batch\_size.

2. Обчислюються градієнти функції втрат відносно параметрів моделі.

3. Параметри оновлюються за допомогою SGD.

**4. Аналіз результатів**

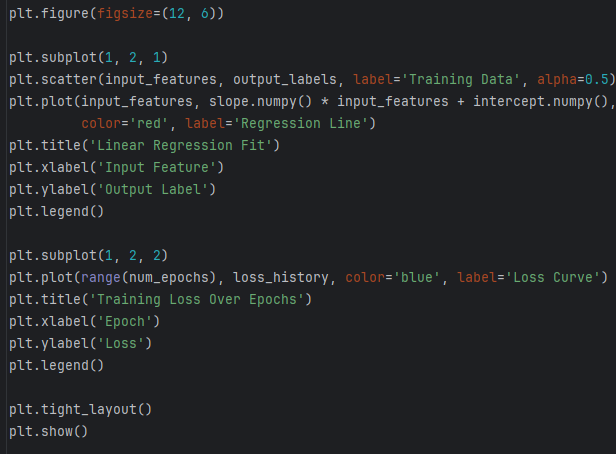
Після завершення навчання модель повертає знайдені параметри лінії регресії: коефіцієнт нахилу kk та вільний член bb.

**Виведення результатів:**

****

Також будується графік, який порівнює вихідні дані та знайдену лінію регресії, а також графік втрат під час навчання.

**Візуалізація результатів:**

****

**5. Висновок**

У результаті роботи алгоритму:

1. Параметри лінійної моделі (kk і bb) знаходяться методом стохастичного градієнтного спуску.

2. Алгоритм поступово знижує втрати, що відображається на графіку зміни функції втрат.

3. Знайдена лінія регресії адекватно апроксимує вихідні дані, незважаючи на доданий шум.

**Приклад результатів**:

Це значення близькі до заданих k=2k=2 та b=1b=1, що підтверджує ефективність алгоритму.

**Висновки:**

в ході виконання лабораторної роботи було досліджено ресурси Keras і TensorFlow. Застосовано TensorFlow.