Лабораторна робота №8

Ресурси Keras. TensorFlow. Навчання лінійної регресії

Виконав: ІП3-21-3 Осипчук Антон Олексійович

Github: https://github.com/AntonOsypchuk1/ai_lab/tree/main/lab8

Завдання. Використовуючи засоби TensorFlow, реалізувати код наведений нижче та дослідити структуру розрахункового алгоритму.

Лістинг програми:

```
import numpy as np
import tensorflow as tf

n_samples, batch_size, num_steps = 1000, 100, 20000
X_data = np.random.uniform(1, 10, (n_samples, 1)).astype(np.float32)
y_data = 2 * X_data + 1 + np.random.normal(0, 2, (n_samples,
1)).astype(np.float32)

X_data = (X_data - np.mean(X_data)) / np.std(X_data)
y_data = (y_data - np.mean(y_data)) / np.std(y_data)

k = tf.Variable(tf.random.normal((1, 1), stddev=0.1), name='slope')
b = tf.Variable(tf.zeros((1,)), name='bias')

optimizer = tf.keras.optimizers.SGD(learning_rate=0.001)

display_step = 100
for i in range(num_steps):
    indices = np.random.choice(n_samples, batch_size)
    X_batch, y_batch = X_data[indices], y_data[indices]

    with tf.GradientTape() as tape:
        y_pred = tf.matmul(X_batch, k) + b
        loss = tf.reduce_sum((y_batch - y_pred) ** 2)

    gradients = tape.gradient(loss, [k, b])
    clipped_gradients = [tf.clip_by_value(g, -1.0, 1.0) for g in
gradients]
    optimizer.apply_gradients(zip(clipped_gradients, [k, b]))

    if (i + 1) % display_step == 0:
        print(f'Epoch {i + 1}: Loss = {loss.numpy():.8f}, k =
{k.numpy()[0][0]:.4f}, b = {b.numpy()[0]:.4f}')
```

Результат виконання:

```
(venv) tensorflow_envantonosypchuk@MacBook-Air-Anton lab8 % python3.11 task1.py
Epoch 100: Loss = 81.96632385, k = 0.0590, b = -0.0003
Epoch 200: Loss = 72.13908386, k = 0.1590, b = -0.0158
Epoch 300: Loss = 63.00621414, k = 0.2590, b = -0.0069
Epoch 400: Loss = 34.90282440, k = 0.3590, b = -0.0096
Epoch 500: Loss = 40.48344803, k = 0.4590, b = -0.0088
Epoch 600: Loss = 33.95607376, k = 0.5590, b = -0.0081
Epoch 700: Loss = 23.00072670, k = 0.6590, b = 0.0018
Epoch 800: Loss = 15.10720444, k = 0.7590, b = 0.0086
Epoch 900: Loss = 16.31402206, k = 0.8578, b = 0.0031
Epoch 1000: Loss = 12.55556774, k = 0.9262, b = 0.0060
Epoch 1100: Loss = 13.49349594, k = 0.9333, b = 0.0016
Epoch 1200: Loss = 13.67977333, k = 0.9346, b = -0.0020
Epoch 1300: Loss = 13.37050819, k = 0.9300, b = 0.0047
Epoch 1400: Loss = 16.18867874, k = 0.9308, b = -0.0041
Epoch 1500: Loss = 11.40747356, k = 0.9265, b = -0.0015
Epoch 1600: Loss = 12.17908382, k = 0.9264, b = -0.0012
Epoch 1700: Loss = 15.22464371, k = 0.9275, b = 0.0041
Epoch 1800: Loss = 18.61181259, k = 0.9227, b = -0.0013
Epoch 1900: Loss = 11.59500885, k = 0.9247, b = -0.0039
Epoch 2000: Loss = 13.51444626, k = 0.9250, b = 0.0027
```

Для кожної ітерації (epoch) обчислюється поточна помилка. На основі цієї помилки оновлюються параметри k та b для зменшення Loss, використовуючи градієнтний спуск.

На початку навчання значення Loss велике (81.96). З кожною ітерацією значення поступово зменшується, досягаючи близько 10-15, що свідчить про покращення моделі.

З кожною ітерацією модель наближається до мінімального значення помилки, а параметри k і b стають точнішими. Параметр k наближається до значення ~0.93-0.94. Параметр b коливається близько 0, що свідчить про те, що його вплив на результат мінімальний або модель знаходить рівновагу в даних.

В результаті виконання програми вдалося побудувати модель, яка поступово зменшує помилку (Loss) та уточнює значення параметрів k і b. Остаточна модель має значення k ~0.93 і b ~0, що відповідає оптимальному наближенню до навчальних даних. Хоча модель не досягла ідеального результату (Loss не став нульовим), вона демонструє стабільне навчання з поступовим покращенням.