Лабораторна робота № 8

Ресурси Keras. TensorFlow. Навчання лінійної регресії

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

Хід роботи

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

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

Лістинг:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow.compat.v1 as tf

tf.disable_v2_behavior()

# Параметри навчання
n_samples = 1000
batch_size = 100
num_steps = 20000
display_step = 1000
learning_rate = 0.0001  # Швидкість навчання

print("Генерація даних...")
X_data = np.random.uniform(1, 10, (n_samples, 1))
y_data = 2 * X_data + 1 + np.random.normal(0, 2, (n_samples, 1)) # y = 2x + 1 + шум

# входи в граф обчислень
X = tf.placeholder(tf.float32, shape=(batch_size, 1), name="Input_X")
y = tf.placeholder(tf.float32, shape=(batch_size, 1), name="Input_y")

# параметри, які модель буде вчити
with tf.variable_scope('linear-regression'):
    k = tf.Variable(tf.random_normal((1, 1)), name='slope')  # Нахил (вага)
    b = tf.Variable(tf.zeros((1,)), name='bias')  # Зсув

# Модель
y_pred = tf.matmul(X, k) + b

# Сума квадратів помилок
loss = tf.reduce_sum((y - y_pred) ** 2)

# Алгоритм градієнтного спуску
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss)

# Запуск сесії
print("Початок навчання...")
```

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

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

        # Запуск кроку оптимізації
        _, loss_val, k_val, b_val = sess.run([optimizer, loss, k, b],
                                             feed_dict={X: X_batch, y: y_batch})

        if (i + 1) % display_step == 0:
            print('Епоха %d: Loss=%.4f, k=%.4f, b=%.4f' % (i + 1, loss_val, k_val, b_val))

    print("\nНавчання завершено!")
    print(f"Фінальні значення: k = {k_val[0][0]:.4f} (очікувалось ~2), b = {b_val[0]:.4f}
(очікувалось ~1)")

    plt.figure(figsize=(10, 6))
    plt.scatter(X_data, y_data, s=10, label='Дані (з шумом)')

    y_line = k_val[0][0] * X_data + b_val[0]
    plt.plot(X_data, y_line, color='red', linewidth=2, label=f'Лінійна регресія:
y={k_val[0][0]:.2f}x + {b_val[0]:.2f}')

    plt.title('Лінійна регресія з TensorFlow')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
    plt.show()
```

Результат:

```
Епоха 1000: Loss=432.5652, k=2.0615, b=0.5984
Епоха 2000: Loss=374.0781, k=2.0311, b=0.5547
Епоха 3000: Loss=418.6169, k=2.0950, b=0.5631
Епоха 4000: Loss=351.8141, k=2.0222, b=0.5599
Епоха 5000: Loss=407.7662, k=2.0341, b=0.5710
Епоха 6000: Loss=281.6876, k=2.0455, b=0.5855
Епоха 7000: Loss=493.9909, k=2.0833, b=0.6033
Епоха 8000: Loss=494.7961, k=1.9992, b=0.5806
Епоха 9000: Loss=444.8780, k=2.0623, b=0.5780
Епоха 10000: Loss=365.1961, k=2.0690, b=0.6046
Епоха 11000: Loss=379.4101, k=2.0687, b=0.5722
Епоха 12000: Loss=391.4243, k=2.0711, b=0.5868
Епоха 13000: Loss=381.2878, k=2.0145, b=0.5771
Епоха 14000: Loss=423.0783, k=2.0598, b=0.6029
Епоха 15000: Loss=370.7720, k=2.0572, b=0.5719
Епоха 16000: Loss=478.3491, k=2.0548, b=0.5919
Епоха 17000: Loss=439.3179, k=2.0207, b=0.5753
Епоха 18000: Loss=375.9739, k=2.0732, b=0.5831
Епоха 19000: Loss=323.9495, k=2.0518, b=0.5894
Епоха 20000: Loss=426.4475, k=2.0468, b=0.5569

Навчання завершено!
Фінальні значення: k = 2.0468 (очікувалось ~2), b = 0.5569 (очікувалось ~1)
```

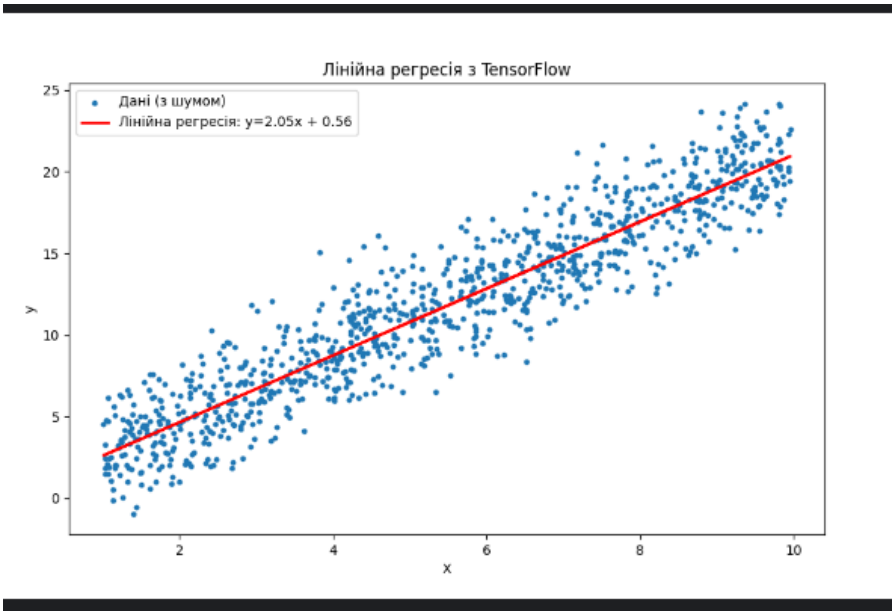

Рис. 1.1-2. Результат дослідження структури розрахункового алгоритму

Результат показує, що алгоритм градієнтного спуску успішно мінімізував функцію втрат. Отримані параметри моделі наблизилися до істинних значень, заданих при генерації даних (y = 2x + 1):

- Розрахунковий коефіцієнт нахилу 2.05 майже ідеально співпав з істинним значенням (2.0).
- Зсув 0.56 дещо відрізняється від істинного (1.0), що пояснюється впливом значного випадкового шуму (з дисперсією 2), доданого до даних.

Експеримент підтвердив працездатність створеного обчислювального графа та ефективність використання TensorFlow для знаходження залежностей у зашумлених даних.

Висновок до роботи: у ході виконання лабораторної роботи було було досліджено бібліотеку TensorFlow та реалізовано алгоритм навчання лінійної регресії. За допомогою засобів TensorFlow було побудовано обчислювальний граф. У результаті навчання методом градієнтного спуску на зашумлених даних модель успішно відновила коефіцієнти лінійної залежності.

Репозиторій: https://github.com/MykolaTrofimchuk/AI-systems/tree/main/Lab08