ЛАБОРАТОРНА РОБОТА № 8

Ресурси Keras. TensorFlow. Навчання лінійної регресії

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

Варіант 1

Хід роботи:

Завдання. Використовуючи засоби TensorFlow, реалізувати код наведений нижче та дослідити структуру розрахункового алгоритму. Для виконання розрахунків, можна використовувати онлайн – середовище google – colab (перехід за посиланням: http://neuralnetworksanddeeplearning.com/chap4.html)

1) спочатку створюємо випадковим чином вхідні дані X_data та y_data за таким алгоритмом: • створюємо 1000 випадкових точок рівномірно на інтервалі [0; 1]; • підраховуємо для кожної точки x відповідну «правильну відповідь» y за формулою y = 2x+1+ϵ, де ϵ - випадково розподілений шум із дисперсією 2, ϵ ~ N(ϵ;0,2);

2) потім оголошуємо tf.placeholder для змінних X та y; на цьому етапі вже потрібно задати їм розмірність, і це в нашому випадку матриця розмірності (розмір міні-батча × 1) для X і просто вектор довжини розмір міні-батча для y;

3) далі ініціалізуємо змінні k та b; це змінні TensorFlow, які поки що жодних значень не мають, але будуть ініціалізовані стандартним нормальним розподілом для k і нулем для b;

4) потім ми встановлюємо власне суть моделі і при цьому будуємо функцію помилки ∑ – 2 (ˆ) i i y y ; зверніть увагу на функцію reduce_sum: на виході вона лише підраховує суму матриці по рядках, але користуватися треба саме нею, а не звичайною сумою або відповідними функціями з numpy, тому що так TensorFlow зможе куди більш ефективно оптимізувати процес обчислень;

5) вводимо змінну optimizer - оптимізатор, тобто власне алгоритм, який підраховуватиме градієнти та оновлюватиме ваги; ми вибрали стандартний оптимізатор стохастичного градієнтного спуску; Тепер нам важливо лише відзначити, що тепер щоразу, коли ми просимо TensorFlow підрахувати значення змінної optimizer, десь за лаштунками відбуватимуться оновлення змінних, від яких залежить оптимізована змінна loss, тобто k і b; по X та y оптимізації не буде, тому що значення tf.placeholder повинні бути жорстко задані, це вхідні дані;

6) записуємо великий цикл, що робить ці оновлення (тобто багато разів обчислює змінну optimizer); на кожній ітерації циклу ми беремо випадкове підмножина з batch_size (тобто 100) індексів даних та підраховуємо значення потрібних змінних; ми подаємо в функцію sess.run список змінних, які потрібно підрахувати (головне - "обчислити" змінну optimizer, інші потрібні тільки для виводу налагодження), і словник feed_dict, в який записуємо значення вхідних змінних, позначених раніше як tf.placeholder.

```
import numpy as np
import tensorflow as tf

rng = np.random.default_rng(12345)
tf.random.set_seed(12345)

features = rng.random((1000, 1), dtype=np.float32)
targets = 2.0 * features + 1.0 + rng.normal(0, 0.1, (1000, 1))

weight = tf.Variable(tf.random.uniform([1], -1.0, 1.0), name="weight")
bias = tf.Variable(tf.zeros([1]), name="bias")

def model(inputs):
    return weight * inputs + bias

sgd_optimizer = tf.optimizers.SGD(learning_rate=0.05)

for iteration in range(20000):
    with tf.GradientTape() as gradient_tape:
        predicted = model(features)
        mse_loss = tf.reduce_mean(tf.square(targets - predicted))

    grads = gradient_tape.gradient(mse_loss, [weight, bias])
    sgd_optimizer.apply_gradients(zip(grads, [weight, bias]))

    if iteration % 1000 == 0:
```

```
        print(
            f"Iteration {iteration}: Loss={mse_loss.numpy():.4f},
Weight={weight.numpy()[0]:.4f}, Bias={bias.numpy()[0]:.4f}"
        )

print(f"\nTrained Parameters: Weight={weight.numpy()[0]:.4f},
Bias={bias.numpy()[0]:.4f}")
```

```
Iteration 0: Loss=2.4758, Weight=1.0035, Bias=0.1539
Iteration 1000: Loss=0.0105, Weight=2.0153, Bias=0.9994
Iteration 2000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 3000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 4000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 5000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 6000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 7000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 8000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 9000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 10000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 11000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 12000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 13000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 14000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 15000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 16000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 17000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 18000: Loss=0.0105, Weight=2.0161, Bias=0.9990
Iteration 19000: Loss=0.0105, Weight=2.0161, Bias=0.9990

Trained Parameters: Weight=2.0161, Bias=0.9990
```

Посилання на Github:

https://github.com/Vladislaw2533/SHI_Barabash_Vlad_IPZ_21_3

Висновки: Дослідив ресурси Keras і TensorFlow. Застосував TensorFlow