Лабораторна робота №8

Тема: «Ресурси Keras. TensorFlow. Навчання лінійної регресії».

Мета: дослідження ресурсу Keras і TensorFlow. Застосування TensorFlow.

Хід роботи:

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

TensorFlow – навчання лінійної регресії. Лінійна регресія - це фактично один нейрон, який отримує на вхід вектор значень x, видає число, і на даних {(x1,y1),(x2,y2),...,(xN,yN)} завдання полягає у тому, щоб мінімізувати суму квадратів відхилень оцінок нейрона і $\hat{y}_i$ істинних значень і y_i.

Наближатимемо лінійною регресією функцію виду f = kx + b для k = 2 і b = 1; k та b будуть параметрами, які ми хочемо навчити.

Лістинг програми:

```
import sys
sys.path.insert(0, r"D:\PythonPackages")

import numpy as np
import tensorflow as tf

print("TensorFlow версія:", tf.__version__)

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)

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

optimizer = tf.keras.optimizers.SGD(learning_rate=0.0001)

display_step = 1000
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 = X_batch * k + b
        loss = tf.reduce_sum((y_batch - y_pred) ** 2)

    gradients = tape.gradient(loss, [k, b])
    optimizer.apply_gradients(zip(gradients, [k, b]))
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.15.000 – Лр.8		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Кохан *Т.О.*			Звіт з лабораторної роботи №8	Літ.	Арк. 1 / Аркушів 3
Перевір.		Маєвський О. В.					
Реценз.						ФІКТ, гр. ІПЗ-22-3	
Н. Контр.							
Зав.каф.		Вакалюк Т.А.					

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

print("Навчання завершено!")
print("Кінцевий k:", k.numpy()[0][0])
print("Кінцевий b:", b.numpy()[0])
```

Результат роботи програми:

```
TensorFlow версія: 2.12.0
Епоха 1000: Втрати=462.1203, k=2.0366, b=0.9138
Епоха 2000: Втрати=564.1190, k=2.0399, b=0.9313
Епоха 3000: Втрати=484.5254, k=2.0170, b=0.9786
Епоха 4000: Втрати=324.6177, k=2.0321, b=1.0108
Епоха 5000: Втрати=338.0840, k=2.0190, b=0.9702
Епоха 6000: Втрати=306.4339, k=2.0326, b=0.9455
Епоха 7000: Втрати=310.7062, k=2.0081, b=0.9978
Епоха 8000: Втрати=290.3992, k=2.0388, b=0.9379
Епоха 9000: Втрати=342.1115, k=2.0466, b=0.9653
Епоха 10000: Втрати=344.0594, k=2.0526, b=0.9502
Епоха 11000: Втрати=415.7527, k=2.0222, b=0.9516
Епоха 12000: Втрати=362.2851, k=2.0538, b=0.9326
Епоха 13000: Втрати=367.6537, k=2.0567, b=0.9430
Епоха 14000: Втрати=309.1408, k=1.9822, b=0.9685
Епоха 15000: Втрати=407.7715, k=1.9918, b=0.9654
Епоха 16000: Втрати=324.0568, k=2.0576, b=0.9497
Епоха 17000: Втрати=352.8323, k=2.0519, b=0.9499
Епоха 18000: Втрати=348.4695, k=2.0306, b=0.9683
Епоха 19000: Втрати=356.0711, k=2.0083, b=0.9763
Епоха 20000: Втрати=343.7823, k=2.0243, b=0.9690
Навчання завершено!
Кінцевий k: 2.0242748
Кінцевий b: 0.96899486
```

Рис. 8.1. Вивід у консоль

Висновок до завдання: модель лінійної регресії успішно навчилася на даних, що генеруються за формулою y = 2x + 1 + шум. Параметри k ≈ 2.02 і b ≈ 0.97 близькі до істинних значень k = 2 і b = 1.

Втрати поступово зменшувалися та стабілізувалися, що свідчить про успішне навчання та здатність моделі узагальнювати дані.

Завдання 8.2. Високорівневий підхід з використанням Keras.

Лістинг програми:

```
import sys
sys.path.insert(0, r"D:\PythonPackages")
import tensorflow as tf
import numpy as np
from tensorflow import keras


model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
model.fit(xs, ys, epochs=500)
print(model.predict(np.array([10.0])))
```

Результат роботи програми:

```
Epoch 493/500
1/1 [==============================] - 0s 3ms/step - loss: 5.3271e-05
Epoch 494/500
1/1 [==============================] - 0s 3ms/step - loss: 5.2178e-05
Epoch 495/500
1/1 [==============================] - 0s 3ms/step - loss: 5.1106e-05
Epoch 496/500
1/1 [==============================] - 0s 4ms/step - loss: 5.0056e-05
Epoch 497/500
1/1 [==============================] - 0s 2ms/step - loss: 4.9029e-05
Epoch 498/500
1/1 [==============================] - 0s 3ms/step - loss: 4.8020e-05
Epoch 499/500
1/1 [==============================] - 0s 3ms/step - loss: 4.7035e-05
Epoch 500/500
1/1 [==============================] - 0s 3ms/step - loss: 4.6069e-05
1/1 [==============================] - 0s 115ms/step
[[18.980196]]
```

Рис. 8.2. Вивід у консоль

Бачимо програма вивела число, дуже близьке до 19.0, що відповідає правильному результату для функції y = 2*10 - 1

Висновок: під час виконання лабораторної роботи було досліджено можливості бібліотек TensorFlow та Keras для побудови й навчання моделей машинного навчання. Реалізовано лінійну регресію як у низькорівневому режимі з використанням TensorFlow, так і у високорівневому підході за допомогою Keras. У процесі експериментів підтверджено швидкість, зручність і ефективність навчання моделей, а також отримано коректні значення параметрів моделі, близькі до теоретичних.

Посилання на git: https://github.com/KokhanTetiana/AI-Systems