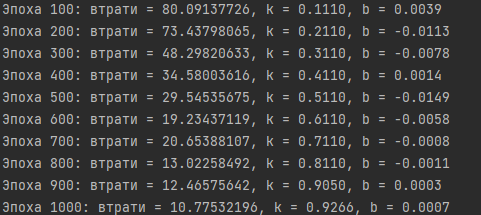
Системи штучного інтелекту.

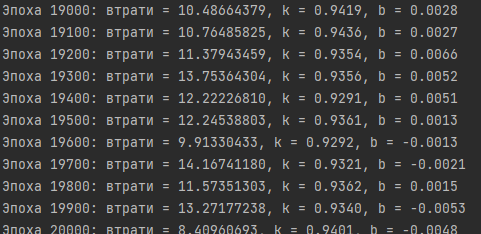
Лабораторна робота 8.Федорович Дмитро ІПЗ-21-3

<https://github.com/Dmitrij3/lab8AI>

**Завдання №1.** Використовуючи засоби TensorFlow, реалізувати код наведений нижче та дослідити структуру розрахункового алгоритму.

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'Эпоха {i + 1}: втрати = {loss.numpy():.8f}, k = {k.numpy()[0][0]:.4f}, b = {b.numpy()[0]:.4f}')

****

****

Кожну епоху(ітерацію) модель знижує помилку до мінімального значення, через те що значення k та b стають точніше. У результаті ми отримали, що на початку втрати були ~80, а у кінці ~8.