Sprawozdanie z przedmiotu
Programowanie Równoległe i Rozproszone

Google Colab

Wykonał:

Imię i nazwisko	Magdalena Paszko
Nr indeksu	76024
Grupa	L2

Prowadzący : dr inż. Krzysztof Szerszeń

1. Wprowadzanie

Usługa Colaboratory (czyli w skrócie „Colab”) umożliwia pisanie i uruchamianie kodu Python bezpośrednio w przeglądarce dzięki:

- brakowi konieczności konfigurowania,
- swobodnemu dostępowi do układów GPU,
- łatwemu udostępnianiu.

Colab może Ci ułatwić pracę niezależnie od tego, czy jesteś studentem, badaczem danych czy badaczem sztucznej inteligencji.

W notatnikach Colab możesz łączyć ze sobą **kod wykonywalny** i **tekst sformatowany** w jeden dokument razem z **obrazami**, **kodem HTML**, **kodem LaTeX** itd. Gdy tworzysz własne notatniki Colab, są one przechowywane na Twoim koncie Dysku Google. Notatniki te możesz z łatwością udostępniać współpracownikom i znajomym, co pozwoli im zamieszczać w nich komentarze, a nawet edytować ich zawartość.

2. Definicja:

a) Całkowanie numeryczne:

Metoda polegająca na przybliżonym obliczaniu całek oznaczonych. Proste metody całkowania numerycznego polegają na przybliżeniu całki za pomocą odpowiedniej sumy ważonej wartości całkowanej funkcji w kilku punktach. Do uzyskania dokładniejszego przybliżenia dzieli się przedział całkowania na niewielkie fragmenty. Ostateczny wynik to suma oszacowań całek w poszczególnych podprzedziałach. Przedział zwykle dzieli się na równe podprzedziały.

b) Interpolacja :

Metoda numeryczna polegająca na budowaniu w danym obszarze $\Omega \subset R^n$, tzw. funkcji interpolacyjnej, która przyjmuje w nim z góry zadane wartości w ustalonych punktach nazywanych węzłami. Stosowana jest zarówno w metodach numerycznych (np. przy obliczaniu całek ze skomplikowanych funkcji), jak i w naukach doświadczalnych przy budowaniu funkcji na podstawie danych pomiarowych w skończonej liczbie punktów (np. w meteorologii przy sporządzaniu map synoptycznych).

3. Implementacja algorytmu:

a) Całkowanie numeryczne metodą prostokątów, trapezów i Simpsona

```python
import math

def func(x):
    return (math.cos((x*x)+0.7)/(1.1+ math.sin((0.6*x)+0.2)))

def metTrapezow(xp,xk,n):
    dx = (xk-xp)/n
    calka =0

    for i in range(1,n):
        calka += func(xp+i*dx)

    calka += (func(xp) + func(xk))/2
    calka *= dx
    return calka


def metSimpsona(xp,xk,n):
    dx = (xk - xp) / n
    calka = 0
    s= 0

    for i in range(1,n):
        x=xp+i*dx
        s+= func(x-dx/2)
        calka += func(x)

    s+= func(xk - dx/2)
    calka = (dx/6)*(func(xp)+ func(xk)+2*calka+4*s)
    return calka
```

```python
def metProstokatow(xp,xk,n):
    dx = (xk-xp)/n
    calka =0
    for i in range(1,n):
        calka += func(xp+i*dx)

    calka *=dx
    return calka



xp = 1.5
xk = 2.8
n = 100000

print("Wynik metoda trapezow wynosi: ",metTrapezow(xp,xk,n))
print("Wynik metoda simpsona wynosi: ",metSimpsona(xp,xk,n))
print("Wynik metoda prostokatow wynosi: ",metProstokatow(xp,xk,n))
```

```
Wynik metoda trapezow wynosi:  0.010253714828639992
Wynik metoda simpsona wynosi:  0.010253714855664982
Wynik metoda prostokatow wynosi:  0.010258925464328213
```

Każda z metod działa na podobnej zasadzie pobierają początek oraz koniec przedziału w jakim maja policzyć całkę wraz z ilością n czyli ilością podzbiorów w jakiej musza obliczyć całkę.

b) Interpolacja

```
import tensorflow as tf
import numpy as np
import logging

logger = tf.get_logger()
logger.setLevel(logging.ERROR)
x = np.array([4.0, 5.0, 6.0, 7.0, 8.0],  dtype=float)
y = np.array([0.0, 18.0, 56.0, 120.0, 216.0],  dtype=float)


l0 = tf.keras.layers.Dense(units=4, activation='relu', input_shape=[1])
l1 = tf.keras.layers.Dense(units=10, activation='relu',)
l2 = tf.keras.layers.Dense(units=1)
model = tf.keras.Sequential([l0, l1, l2])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-1), loss='mean_squared_error', metrics=['mean_squared_error'])

history = model.fit(x, y, epochs=1000, verbose=1)


print("Dla x = 4 (norm = 0 ), y = {} ".format(model.predict([4])))
print("Dla x = 5 (norm = 18), y = {} ".format(model.predict([5])))
print("Dla x = 6 (norm = 56), y = {} ".format(model.predict([6])))
print("Dla x = 7 (norm = 120), y = {} ".format(model.predict([7])))
print("Dla x = 8 (norm = 216), y = {} ".format(model.predict([8])))


print("Model przewiduje to dla x = 15, y = {} ".format(model.predict([15])))
```

Program oblicza funkcje interpolacyjna jest to rozwiniecie programu z zmiana wzoru funkcji i dodaniem nowych warstw dzięki czemu jest to teraz program dla wielowarstwowych sieci neuronowych z jednym wejściem i jednym wyjściem za pomocą biblioteki keras.
Biblioteka Keras Python do głębokiego uczenia się koncentruje się na tworzeniu modeli jako sekwencji warstw.
Dane wejściowe w modelu:
Pierwsza warstwa w modelu musi określać kształt danych wejściowych.
Jest to liczba atrybutów wejściowych i jest definiowana przez argument input_shape. Ten argument oczekuje liczby całkowitej.
Po zdefiniowaniu modelu należy go skompilować. Tworzy to wydajne struktury używane przez bazowy backend (Theano lub TensorFlow) w celu wydajnego wykonywania modelu podczas szkolenia. Kompilujesz swój model za pomocą funkcji compile ()
Następnie jest używane model.fit tutaj najpierw podajemy dane szkoleniowe (X) i etykiety szkoleniowe (Y). Następnie używamy Keras, aby umożliwić naszemu modelowi trenowanie przez 1000 epok. Na koniec wywoływane jest model.predict () służy aby wygenerować wyjście sieciowe dla danych wejściowych.

Wynik:

```
Epoch 997/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.0984 - mean_squared_error: 0.0984
Epoch 998/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.0971 - mean_squared_error: 0.0971
Epoch 999/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.0960 - mean_squared_error: 0.0960
Epoch 1000/1000
1/1 [==============================] - 0s 4ms/step - loss: 0.0949 - mean_squared_error: 0.0949
Dla x = 4 (norm = 0 ), y = [[0.17660728]]
Dla x = 5 (norm = 18), y = [[18.003761]]
Dla x = 6 (norm = 56), y = [[56.299114]]
Dla x = 7 (norm = 120), y = [[120.35191]]
Dla x = 8 (norm = 216), y = [[215.81335]]
Model przewiduje to dla x = 15, y = [[889.79114]]
```

```c
if (ID == 0) {
    printf("\nNr iteracji: %d, Gra w życie:\n", ktoraIteracja);
    for (j = 0; j < ROZMIAR; j++) {
        if (j % SZEROKOSC == 0)
            printf("\n");

        printf("%d  ", plansza[j]);
    }
    printf("\n");
}
```