Міністерство освіти та науки України
Львівський національний університет імені Івана Франка

Лабораторна робота №2

на тему: «Векторизація даних»

Виконав:
студент групи ФеС-31
Козак Дмитро
Перевірив:
Рибак А. В.

Львів – 2021

Мета: Дослідити базові методи та провести аналіз особливостей бібліотеки NumPy на основі реалізованих методів в порівнянні Python vs Python + NumPy.

Хід роботи:

1. Дослідити базові методи бібліотеки NumPy. Навести кілька прикладів у звіті
2. Реалізувати наступні методи без використання бібліотеки NumPy: множення двох матриць (N x M), множення матриці (N x M) на вектор (N), множення вектора (N) на матрицю (N x M) та множення двох векторів (N).
   - N є в межах [100 - 10000];
   - M є в межах [100 - 10000];
   - Матриці та вектори ініціалізуємо випадковими даними в межах [0; 1];
3. Реалізувати методи, які описані в пункті 2, з використанням бібліотеки NumPy.
4. Написати Unit tests для методів.
5. Провести аналіз особливостей бібліотеки NumPy на основі реалізованих методів в порівнянні Python vs Python + NumPy. Характеристика аналізу – час виконання методу.

Результати роботи

1. Дослідив базові методи бібліотека NumPy:

- numpy.empty(*shape*, *dtype=float*, *order='C'*, *, *like=None*) – Створює пустий масив/матрицю з заданим розміром.
- numpy.identity(*n*, *dtype=None*, *, *like=None*) — Створює одиничну матрицю з заданим розміром
- numpy.zeros(*shape*, *dtype=float*, *order='C'*, *, *like=None*) — Створює нульовий масив/матрицю з заданим розміром
- numpy.array(*object*, *dtype=None*, *, *copy=True*, *order='K'*, *subok=False*, *ndmin=0*, *like=None*) — створює новий масив

2. Для того щоб реалізувати множення матриць та векторів без використання NumPy було створено клас Matrix(замість вектора буде використовуватись матриця з розмірністю (n, 1) або (n, 1)). У ньому було передбачено методи:

- __init__(self, matrix) – магічний метод для ініціалізації об’єкта з заданою матрицею у форматі списку зі списків
- generate_random_matrix(cls, shape) – метод класу для створення нової матриці із заданим розміром заповненої випадковими числами
- multiply_rows(row1, row2) – статичний метод для перемноження двох рядків матриці
- to_numpy(self) – метод для перетворення матриці в масив numpy
- __getitem__(self, item) – магічний метод для зручного діставання списків з матриці
- __mul__(self, other) – магічний метод для множення матриць
- __str__(self) – магічний метод для зручного виведення матриці

```
def __mul__(self, other: 'Matrix'):
    shape = (self._shape[0], other._shape[1])
    result = [[0 for _ in range(shape[1])] for _ in range(shape[0])]
    if isinstance(other, Matrix):
        for i in range(shape[0]):
            for j in range(shape[1]):
                result[i][j] = Matrix.multiply_rows(self[i], other[:, j])
        return Matrix(result)
```

Рис. 1 Метод для множення матриць

3. Для реалізації множення матриць та векторів з допомогою бібліотеки NumPy було використано метод numpy.dot який виконує множення матриць

```
def numpy_multiply(matrix1, matrix2):
    return np.dot(matrix1, matrix2)
```

Рис 2. Множення матриць та векторів з допомогою бібліотеки NumPy

5. Виміри часу виконання множення двох матриць з допомогою NumPy та без

1. Створив дві матриці розміром (500, 600) і (600, 500) заповнені випадковими числами в межах [0; 1], перетворив їх у масив numpy та присвоїв змінним.

```
matrix1 = Matrix.generate_random_matrix((500, 600))
matrix2 = Matrix.generate_random_matrix((600, 500))
np_matrix1 = matrix1.to_numpy()
np_matrix2 = matrix2.to_numpy()
```

2. За допомогою функції perf_counter з бібліотеки time провів виміри часу для множення матриць з допомогою NumPy та без

```
start = perf_counter()
result = matrix1 * matrix2
print("Without numpy: {}".format(perf_counter()-start))
start = perf_counter()
result_with_np = numpy_multiply(np_matrix1, np_matrix2)
print("With numpy: {}".format(perf_counter()-start))
```

3. Вивів результати в консолі

```
Without numpy: 33.7011296
With numpy: 0.11091909999999672
```

Як видно з рисунку вище, час виконання методу для множення матриць без використання NumPy був в 3000 разів повільнішим ніж з використанням NumPy

ВИСНОВОК

Під час виконання лабораторної роботи я дослідив базові методи бібліотеки NumPy та провів аналіз її особливостей на основі реалізованих методів в порівнянні Python vs Python + NumPy, в результаті чого виявилось, що методи бібліотеки NumPy працюють значно швидше. Код на звіт завантажив на свій github: https://github.com/newbeepi/LNU_ML.