методів дає змогу використовувати бібліотеку для аналізу даних та прогнозування різноманітних явищ. Їх легко інтегрувати з іншими бібліотеками Python, як-от Matplotlib та Seaborn, що допомагає створювати візуалізації результатів та аналізу даних [3].

Сучасні фреймворки машинного навчання відкривають нові можливості для розробників та дослідників, даючи змогу їм ефективно впроваджувати та вдосконалювати моделі. Вибір конкретного фреймворка залежить від завдань, типів моделей, а також особистих вподобань. Загальна тенденція полягає в тому, що розвиток машинного навчання значною мірою зумовлений появою потужних та гнучких фреймворків, які продовжують визначати та трансформувати цю захоплюючу галузь.

Список використаних джерел

1. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. URL: https://www.tensorflow.org/about/bib (дата звернення: 06.12.2023).

2. Introduction to TensorFlow. URL: https://www.tensorflow.org/learn (дата звернення: 06.12.2023).

3. Scikit-learn: machine learning in Python – scikit-learn 1.3.2 documentation. URL: https://scikit-learn.org/stable/index.html (дата звернення: 06.12.2023).

УДК 004.021

Бурківський О. С., здобувач 2 курсу спеціальності 122 Комп'ютерні науки,

Потапова Н. А., канд. екон. наук, доцент, доцент кафедри інформаційних технологій

ЗАСТОСУВАННЯ ДИНАМІЧНОГО ПРОГРАМУВАННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ПРО НАЙДОВШУ ПОСЛІДОВНІСТЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Задача про найдовшу послідовність є однією з класичних задач алгоритмізації, яка виникає в різних контекстах, як-от обробка рядків, генетика, аналіз текстів тощо. Через це використання методів динамічного програмування може суттєво полегшити розв'язання проблеми та забезпечити ефективний алгоритм для пошуку найдовшої послідовності.

Динамічне програмування – це ефективний метод розв'язання складних задач, який має бути застосований у різних галузях, зокрема алгоритмізації, оптимізації та штучному інтелекті. Основні ідеї динамічного програмування включають:

1. Розбиття задачі на підзадачі. Динамічне програмування передбачає розбиття складної задачі на простіші підзадачі, що сприяє полегшенню розв'язання задачі, шляхом зменшення та більшої керованості.

2. Зберігання та використання попередніх результатів. Ключовим елементом динамічного програмування є збереження результатів розв'язку підзадач для подальшого використання, що дає змогу уникнути зайвих повторних обчислень та підвищити ефективність алгоритму.

3. Визначення рекурентних відносин. Динамічне програмування використовує рекурентні відносини для опису взаємозв'язків між задачами різного розміру. Це означає, що розв'язання більшої задачі може бути ефективно побудоване на основі розв'язань менших підзадач.

4. Визначення базових випадків. Для кожної задачі, що розв'язується динамічним програмуванням, визначаються базові випадки – найпростіші випадки, які можна вирішити без подальших рекурсивних викликів. Базові випадки відіграють ключову роль у рекурсивному розв'язанні задачі.

5. Побудова оптимального розв'язку знизу вгору. Динамічне програмування може використовуватися для побудови оптимального розв'язку знизу вгору, тобто розв'язання менших підзадач та поступове сходження до більшої задачі. Цей підхід дає змогу ефективно обчислити розв'язок для всіх можливих комбінацій підзадач.

6. Ітераційний підхід і таблиці для оптимізації. Динамічне програмування може бути реалізоване як ітераційний алгоритм, який використовує таблиці для збереження результатів обчислень підзадач. Це забезпечує оптимізацію в плані часу та пам'яті.

Задача знаходження найдовшої зростаючої послідовності (LIS) виникає в різних областях, як-от аналіз даних, оптимізація та комп'ютерна наука. Мета полягає в тому, щоб знайти найбільшу послідовність чисел у масиві, де кожне число більше від попереднього. Приклад коду на мові програмування C# для задачі знаходження найдовшої зростаючої послідовності (LIS) з допомогою динамічного програмування (рис. 1).

Постановка задачі: існує масив чисел arr довжиною n. Необхідно визначити найдовшу послідовність індексів i_1, i_2, …..i_k, де $0 <= i_1 < i_2 <... < i_k < n$, таку, що $arr[i_1] < arr[i_2] <... < arr[i_k]$. Довжина цієї найдовшої зростаючої підпослідовності є ключовим показником ефективності розв'язання задачі. Приклад виконання програми наведено на рис. 2.

Тобто якщо вхідний масив чисел {10, 22, 9, 33, 21, 50, 41, 60, 80}, то найбільша зростаюча послідовність у цьому масиві – {10, 22, 33, 50, 60, 80}.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LongestIncreasingSubsequence
{
    class Program
    {
        static int LongestIncreasingSubsequenceLength(int[] arr, int n)
        {
            int[] lis = new int[n];
            int maxLisLength = 1;

            for (int i = 0; i < n; i++)
            {
                lis[i] = 1;

                for (int j = 0; j < i; j++)
                {
                    if (arr[i] > arr[j] && lis[i] < lis[j] + 1)
                    {
                        lis[i] = lis[j] + 1;
                    }
                }

                if (maxLisLength < lis[i])
                {
                    maxLisLength = lis[i];
                }
            }

            return maxLisLength;
        }

        static void Main()
        {
            int[] sequence = { 10, 22, 9, 33, 21, 50, 41, 60, 80 };
            int n = sequence.Length;

            int result = LongestIncreasingSubsequenceLength(sequence, n);
            Console.OutputEncoding = System.Text.Encoding.UTF8;
            Console.WriteLine($"Довжина найдовшої зростаючої підпослідовності становить {result}");
        }
    }
}
```

Рисунок 1. Приклад коду

```
Довжина найдовшої зростаючої підпослідовності становить 6
```

Рисунок 2. Приклад виконання

Отже, знаходження найдовшої зростаючої послідовності з допомогою динамічного програмування використовується для оптимізації торгових стратегій, аналізу даних, а також у задачах, пов’язаних із послідовністю подій чи елементів у великих наборах даних.

Список використаних джерел

1. Задачі динамічного пошуку. URL: http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/zadachi-dynamichnoho-prohramuvannja.pdf (дата звернення: 29.11.2023).

2. Застосування динамічного програмування. URL: https://ua5.org/osnprog/1907-dynamichne-programuvannya.html (дата звернення: 29.11.2023).