Постановка задачі

Стохастичним аналогом методу можливих напрямків (див. файл "Lab_2") розв'язати задачу із файлу "zavd_lab2".

Номер студента у списку групи = варіант задачі.

Не виключено, що задача може не мати розв'язку.

1. Андрусенко Максим Олександрович
2. Березяк Марта Ігорівна
3. Борецька Віталія Юріївна
4. Далекорій Кирил Ігорович
5. Дикий Назар Ігорович
6. Добровольський Назар Євгенович
7. Гудзеляк Наталія Ігорівна
8. Канафоцький Юрій Романович
9. Коваль Данило Петрович
10. Кондера Святослав Юрійович
11. Костецький Юрій Романович
12. Крицький Антон Олександрович
13. Мацько Соломія Олексіївна
14. Подвірний Олег Богданович
15. Рутар Олег Олександрович
16. Саган Софія Ігорівна
17. Спасник Юрій Романович
18. Трущак Максим Ярославович
19. Шевчук Маріан-Северин Андрійович
20. Юськевич Юлія Сергіївна

Оскільки згідно вище вказаного списку групи я 6, але через неправильний порядок прізвищ (Гудзеляк після прізвищ на Д), то моє завдання:

$$
\begin{aligned}
(7)\quad & f_0(x) = 2x_1^2 + 3x_2^2 \to \min, \\
& 3x_1 + x_2 \le 12, \\
& x_1 + 2x_2 \ge 10, \\
& x_1 \ge 0, x_2 \ge 0.
\end{aligned}
$$

Практична реалізація

game theory > lab2.py > ...

```python
import numpy as np

# Objective function
def f_0(x):
    x1, x2 = x
    return 2 * x1**2 + 3 * x2**2

# Gradient of the objective function
def gradient_f_0(x):
    x1, x2 = x
    grad_x1 = 4 * x1
    grad_x2 = 6 * x2
    return np.array([grad_x1, grad_x2])

# Constraints
def constraint1(x):
    return 3 * x[0] + x[1] - 12

def constraint2(x):
    return -x[0] - 2 * x[1] + 10

# Stochastic Gradient Descent
def stochastic_gradient_descent(initial_point, learning_rate, num_iterations):
    x = np.array(initial_point)

    for i in range(num_iterations):
        grad = gradient_f_0(x)
        x1_update = x[0] - learning_rate * grad[0]
        x2_update = x[1] - learning_rate * grad[1]
        x = [max(0, x1_update), max(0, x2_update)]  # Project onto the feasible region

        # Project onto the constraint regions
        if constraint1(x) > 0:
            x[1] = 12 - 3 * x[0]
        if constraint2(x) > 0:
            x[1] = (10 + x[0]) / 2

    return x

# Set initial point, learning rate, and number of iterations
initial_point = [0, 0]
learning_rate = 0.01
num_iterations = 1000

# Run SGD
optimal_solution = stochastic_gradient_descent(initial_point, learning_rate, num_iterations)

# Print the optimal solution
print("Optimal solution:")
print("x1 =", optimal_solution[0])
print("x2 =", optimal_solution[1])
print("Minimum value of f_0(x) =", f_0(optimal_solution))

```

Результат

```
(air) (base) outjack@MacBook-Pro-Nazar lnu % /Users/outjack/opt/anaconda3/envs/air/bin/python "/Users/outjack/Documents/lnu/game theory/lab2.py"
Optimal solution:
x1 = 0
x2 = 5.0
Minimum value of f_0(x) = 75.0
```

Висновок

Вищеописаний Python код є прикладом застосування стохастичного градієнтного спуску (SGD) для вирішення задачі оптимізації як з цільовою функцією, так і з обмеженнями лінійної нерівності.

Деякі коментарі стосовно коду:

1. Код визначає конкретну квадратичну цільову функцію, f_0(x) = 2 * x1^2 + 3 * x2^2, і обчислює її градієнт. Градієнт необхідний для алгоритму SGD для ітеративної мінімізації функції.
2. До задачі введено два обмеження лінійної нерівності: 3 * x1 + x2 ≤ 12 і x1 + 2 * x2 ≥ 10. Ці обмеження обмежують можливу область, у якій алгоритм має знайти оптимальне рішення.
3. Функція stochastic_gradient_descent виконує процес оптимізації SGD. Вона починається з початкової точки, оновлює точку за допомогою градієнта та гарантує, що точка залишається в межах можливої області, проектуючи її на межі обмежень. Ця проекція гарантує, що алгоритм поважає обмеження, шукаючи мінімум цільової функції.
4. Код дозволяє налаштувати початкову точку, швидкість навчання та кількість ітерацій. Тобто, користувач має змогу контролювати збіжність і точність процесу оптимізації.
5. Після завершення оптимізації код отримує оптимальне рішення та обчислює мінімальне значення цільової функції. Потім ці результати друкуються для перегляду користувачем.

Використана література

1. https://realpython.com/gradient-descent-algorithm-python/
2. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/