Засоби підготовки та аналізу даних

Лабораторна робота 6

ФБ-24 Воровська Єва

Посилання на репозиторій (GitHub repository):
https://github.com/yevavorov/ad_labs.git

Мета роботи: отримати поглиблені навички роботи з numpy; дослідити поняття лінійної регресії та градієнтного спуску.

Постановка задачі

Ознайомтесь з теоретичним матеріалом. Створіть програму для обчислення лінійної регресії методом найменших квадратів та градієнтним спуском.

Завдання 1

1. Згенеруйте двовимірні дані `(x, y)` за допомогою `numpy.random`: бажано, щоб розподіл точок був навколо деякої наперед заданої прямої `(y = kx + b)` для подальшого аналізу результатів.

2. Напишіть функцію, яка реалізує метод найменших квадратів для пошуку оптимальних оцінок $\hat{k}$ та $\hat{b}$.

3. Порівняйте знайдені параметри з оцінкою `np.polyfit(x,y,1)` (оцінка полінома степеню 1 методом найменших квадратів), та з початковими параметрами прямої (якщо такі є).

4. Відобразіть на графіку знайдені оцінки лінії регресії (вашої та `numpy`). Якщо ви генерували вхідні дані навколо лінії, відобразіть також її.

[laba6_1]

Код:

`[Так як другий код по суті є доповненням до першого (цього), то` *`{див. далі}`*`]`

Завдання 2

1. Напишіть функцію, яка реалізує метод градієнтного спуску для пошуку оптимальних оцінок $\hat{k}$ та $\hat{b}$. Визначіть оптимальні вхідні параметри: `learning_rate, n_iter`

2. Додайте отриману лінію регресії на загальний графік

3. Побудуйте графік похибки від кількості ітерацій, зробіть висновки

4. Порівняйте отримані результати з результатами попереднього завдання

[laba6_2]

Код:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import TextBox, CheckButtons, Slider
# ^ Завантаження бібліотек ^


# Середньоквадратична похибка
def mean_squared_error(x, y, k, b):
    y_pred = k*x + b
    mse = np.mean((y - y_pred) ** 2)
    return mse

# Градієнтний спуск
def gradient_descent(x, y, learning_rate, n_iter):
    k = 0
    b = 0
    n = len(x)
    mse_history = []
    for _ in range(n_iter):
        y_pred = k*x + b
        k_grad = (-2/n) * np.sum(x * (y - y_pred))
        b_grad = (-2/n) * np.sum(y - y_pred)
        k -= learning_rate * k_grad
        b -= learning_rate * b_grad
        mse = mean_squared_error(x, y, k, b)
        mse_history.append(mse)
    return k, b, mse_history

# y = kx + b
def generate_data(num_points, k, b, noise_level):
    x = np.linspace(0, 10, num_points)
    noise = np.random.normal(0, noise_level, num_points)
```

```
    y = k*x + b + noise
    return x, y

# Метод найменших квадратів (МНК)
def mnk(x, y):
    x_mean = np.mean(x)
    y_mean = np.mean(y)
    xy_mean = np.mean(x*y)
    x_sq_mean = np.mean(x**2)
    k = (xy_mean - x_mean*y_mean) / (x_sq_mean - x_mean**2)
    b = y_mean - k*x_mean
    return k, b

# Згенеровані дані
learning_rate = 0.01
n_iter = 1000
true_k = 2
true_b = 5
num_points = 100
noise_level = 2
x, y = generate_data(num_points, true_k, true_b, noise_level)
grad_k, grad_b, mse_history = gradient_descent(x, y, learning_rate,
n_iter)

# Оцінки параметрів k та b (наша функція ↑ та np.polyfit)
mnk_k, mnk_b = mnk(x, y)
polyfit_k, polyfit_b = np.polyfit(x, y, 1)

# Результати
text_results = f"""Метод найменших квадратів (МНК):
    k: {mnk_k}
    b: {mnk_b}

np.polyfit:
    k: {polyfit_k}
    b: {polyfit_b}

Градієнтний спуск:
    k: {grad_k}
    b: {grad_b}

Початкова пряма:
    k: {true_k}
    b: {true_b}"""

fig, ax = plt.subplots(1, 2, figsize=(12, 8))
fig.subplots_adjust(top=0.63, left=0.1, bottom=0.08)
```

```
# TextBox для результутів
ax_text = plt.axes([0.1, 0.64, 0.6, 0.32])
text_box = TextBox(ax_text, "Results:", initial=text_results)

# Checkbox для ліній
ax_MNK = plt.axes([0.71, 0.7, 0.19, 0.26])
check_lines = CheckButtons(ax_MNK, ["МНК", "np.polyfit", "Град. спуск"],
[True, True, True])

# Slider для n_iter
ax_n_iter = plt.axes([0.75, 0.64, 0.15, 0.05])
s_n_iter = Slider(ax_n_iter, "n_iter", valmin=100, valmax=1000,
valstep=100, valinit=n_iter)

# Update (оновлення параметрів)
def update(val):
    MNK_status = check_lines.get_status()[0]
    polyfit_status = check_lines.get_status()[1]
    grad_status = check_lines.get_status()[2]
    new_n_iter = s_n_iter.val
    new_grad_k, new_grad_b, new_mse_history = gradient_descent(x, y,
learning_rate, new_n_iter)
    linereg.set_ydata(new_grad_k*x + new_grad_b)
    new_p = f"[{new_n_iter}] Похибка: {new_mse_history[new_n_iter-1]}"
    p_text.set_text(new_p)
    new_text_results = f"""Метод найменших квадратів (МНК):
    k: {mnk_k}
    b: {mnk_b}

np.polyfit:
    k: {polyfit_k}
    b: {polyfit_b}

Градієнтний спуск:
    k: {new_grad_k}
    b: {new_grad_b}

Початкова пряма:
    k: {true_k}
    b: {true_b}"""
    text_box.set_val(new_text_results)
    if MNK_status:
        lineMNK.set_visible(True)
    else:
        lineMNK.set_visible(False)
    if polyfit_status:
```

```
        linepoly.set_visible(True)
    else:
        linepoly.set_visible(False)
    if grad_status:
        linereg.set_visible(True)
    else:
        linereg.set_visible(False)
    fig.canvas.draw_idle()

check_lines.on_clicked(update)
s_n_iter.on_changed(update)


# Графік
ax[0].scatter(x, y, color="black", label="Дані")
line, = ax[0].plot(x, true_k*x + true_b, color="red", label="Пряма")
lineMNK, = ax[0].plot(x, mnk_k*x + mnk_b, color="green", label="МНК")
linepoly, = ax[0].plot(x, polyfit_k*x + polyfit_b, color="blue",
label="np.polyfit")
linemse, = ax[1].plot(range(1, n_iter + 1), mse_history, color="magenta")
linereg, = ax[0].plot(x, grad_k*x + grad_b, color="orange", label="Град.
спуск")
ax[0].legend()
ax[0].grid(color="0.8")
p = f"[{n_iter}] Похибка: {mse_history[n_iter-1]}"
p_text = ax[1].text(150, 32, p, fontsize=12)
plt.show()
```