수치해석 HW13

2018008613 안상욱

1. Overview

- 기존의 Line은 $y = 2x - 1$이고, x의 범위는 -5부터 6까지 12개의 정수를 주었습니다. 이에 해당하는 12개의 점을 $y = 2x - 1 + N(0, 2)$ 를 이용해 각각의 12개의 x값에 해당하는 y값을 구해 12개의 점을 생성했습니다.
- 12개의 점 중 6개의 점을 랜덤하게 선정해서 Least square를 이용한 fitting을 10번 진행했고 이 중에서 가장 cost가 작았을 때의 직선으로 fitting했습니다.
- 그 후 12개의 점 을 모두 가지고 least squar를 해서 직선을 fitting한 후 두 직선을 비교해 보았습니다.

2. 코드

- 먼저 gaussian_noise라는 함수를 만들었는데, 이는 초기 y값을 구할 때 N(0,2)에 해당하는 noise를 만들어 주기 위한 함수입니다.
- 12개의 점에 해당하는 x, y 좌표를 dx, dy에 저장해주었고, dy에는 gaussian noise 값을 더해주었습니다.
- A, b 행렬은 12개의 점 중 6개를 뽑아서 Least square를 하기 위한 행렬입니다.
- 그리고 a, c는 Least square를 통해 fitting한 line이 y = ax + c 형태를 가질 때의 상수값이고, cost는 10번의 시행을 하면서 각각의 line들에 대한 오차값을 이용해 가장 오차값이 작을 때의 a, c값을 a, c 변수에 저장해주었습니다.

```
def gaussian_noise(x):
    return (1 / np.sqrt(2 * np.pi * 2)) * np.exp(- x ** 2 / 4)

dx = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6]
dy = [-11, -9, -7, -5, -3, -1, 1, 3, 5, 7, 9, 11]
for i in range(12):
    dy[i] += gaussian_noise(dx[i])

A = np.zeros((6,2))
b = np.zeros(6)

for i in range(6):
    A[i][1] = 1

a = 0
c = 0
cost = 999999999
```

2. 코드

- 10번 6개의 랜덤한 좌표를 골라서 Least square를 이용해 line fitting을 해 주었고 cost, 즉 error가 가장 작았던 line을 y = ax + c라고 했을 때 이 상수 a와 c를 각각 a와 c에 저장해 주었습니다.

```
for t in range(10):
    x = list(range(12))
    x_train, x_test = train_test_split(x, test_size=0.5)

    for i in range(6):
        A[i][0] = dx[x_train[i]]
        b[i] = dy[x_train[i]]

    x = np.linalg.inv(A.T@A)@A.T@b

    this_cost = 0
    b1 = A @ x
    for i in range(6):
        this_cost += (b1[i]-dy[x_train[i]]) * (b1[i]-dy[x_train[i]])
    if cost > this_cost:
        a = x[0]
        c = x[1]
        cost = this_cost
```

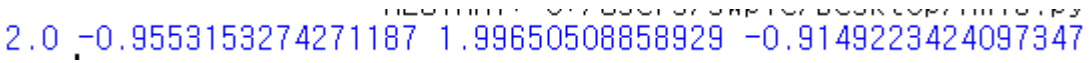
2. 코드

- 12개의 좌표를 모두 가지고 Least square를 통해 line fitting한 결과 y = a1 * x + c1의 line을 갖게 되는데 이 a1과 c1을 각각 a1과 c1에 저장한 뒤 a, c, a1, c1값을 모두 출력해 보았습니다.

```
A = np.zeros((12,2))
b = np.zeros(12)

for i in range(12):
    A[i][1] = 1
    A[i][0] = dx[i]

b = dy

x = np.linalg.inv(A.T@A)@A.T@b
a1 = x[0]
c1 = x[1]
print(a, c, a1, c1)
```

3. 실행 결과

- 다음과 같이 a = 2.0, c = -0.955, a1 = 1.997, c1 = -0.915의 값을 갖는 것을 확인해 볼 수 있었습니다.
- 즉, RANSAC을 이용해 6개의 랜덤한 좌표를 가지고 Line fitting 했을 때의 함수는 y = 2.0x – 0.955이고,
- 12개의 좌표를 모두 사용해 Line fitting 했을 때의 함수는 y = 1.997x - 0.915라는 값을 가짐을 확인해 볼 수 있었습니다.



4. 결과 분석

- RANSAC을 이용해 6개의 랜덤한 좌표를 가지고 Line fitting 했을 때의 함수는 y = 2.0x – 0.955이고,
- 12개의 좌표를 모두 사용해 Line fitting 했을 때의 함수는 y = 1.997x - 0.915라는 값을 가짐을 확인해 볼 수 있었습니다.
- 12개를 모두 이용해서 line fitting 한 결과보다 RANSAC을 이용해 랜덤한 6개의 좌표만을 가지고 Line fitting 한 결과가 더 정확하게 나오는 것을 확인해 볼 수 있었습니다.
- 이러한 결과가 나오는 이유는 12개의 좌표에는 noise가 섞여 있기 때문에 12개를 모두 이용한다면 더 많은 noise가 섞이므로 RANSAC 을 이용했을 때보다 부정확한 결과가 나오는 것을 확인해 볼 수 있었습니다.