수치해석
HW #13

Line fitting using RANSAC and least square

2018007956 김채아

[Line fitting 과정] x: [−5,6] 정수 범위에서 y=2x−1 함수의 좌표들을 사용한다

```
# y = 2x-1
Points = [[-5,-11.0], [-4,-9], [-3,-7], [-2,-5], [-1,-3], [0,0], [1,1], [2,3], [3,5], [4,7], [5,9], [6,11]]
Points = np.array(Points)
```

〈When using whole samples〉

❑ How to generate 12 samples

- Line: y = 2x−1 ← 정답
- x: [−5, +6], integer
  - ➢ −5, −4, −3, ⋯..5, 6
- y=2x−1 + N(0, 2)
  - ➢ Adding Gaussian noise for each y = 2x−1

mean = 0, variance = 2 인 가우시안 노이즈를 랜덤하게 생성해서 더함
→ 이값은 규칙적으로 바뀌는데 실값은 랐다갔다
《이렇게 12개의 좌표 생성》

❑ Due

- 2020. 12. 2

〈When using RANSAC〉

❑ Line fitting using RANSAC and least square

- 1. Randomly select 6 samples from 12 points
- 2. Find the fitted line using 6 samples and least square
- 3. Calculate the error
- 4. Compare the current error with previous error
  - ➢ If current error is smaller than the previous one, save the current solution and remove the previous one
- 5. Repeat 1~4 until your criterion

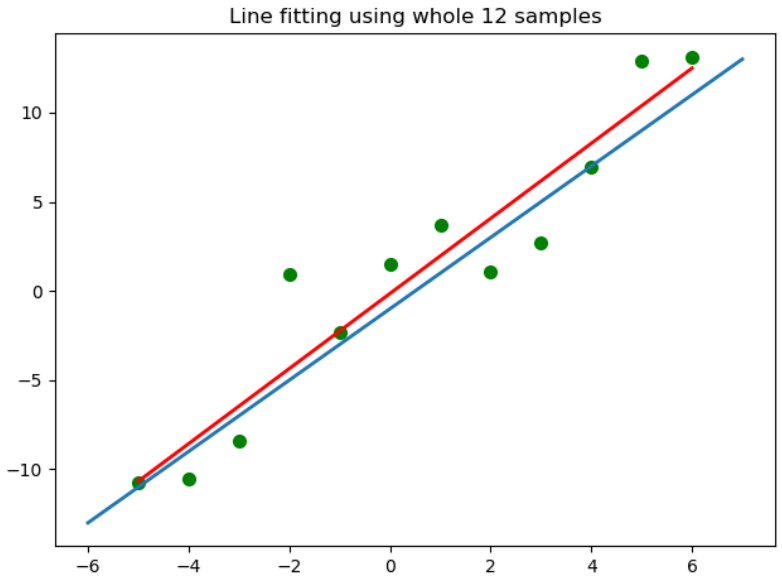
❑ Compare the solution of RANSAC with the solution of whole 12 samples

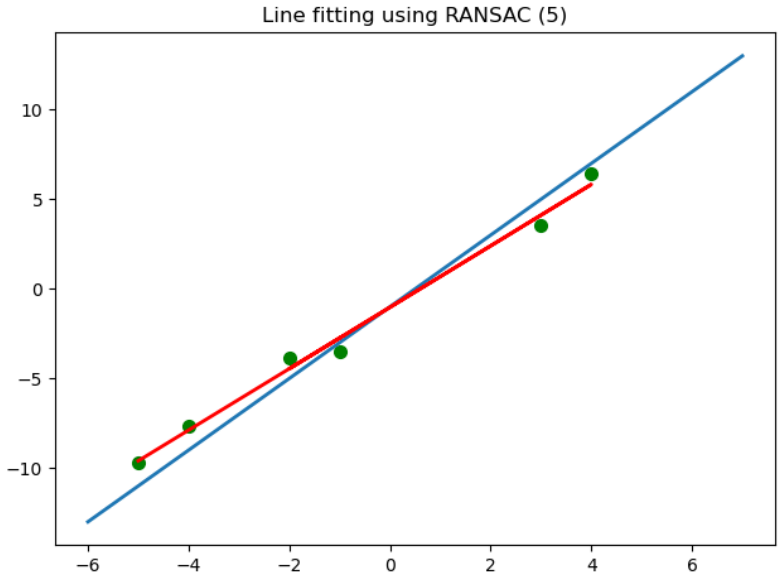
1. 평균=0, 분산=2인 가우시안 노이즈를 랜덤하게 생성해서 y값에 더해 12개의 샘플을 만든다
2. 노이즈 좌표들의 least square로 line fitting 한다
3. y=2x−1과 차이를 계산해본다

1. 랜덤하게 6개의 샘플을 고르고 가우시안 노이즈를 추가한다
2. least square로 line fitting한다
3. y=2x−1과 차이(에러)를 계산해본다
4. 특정 조건이 만족할 때까지 앞의 과정들을 반복하며 최소 에러를 구한다

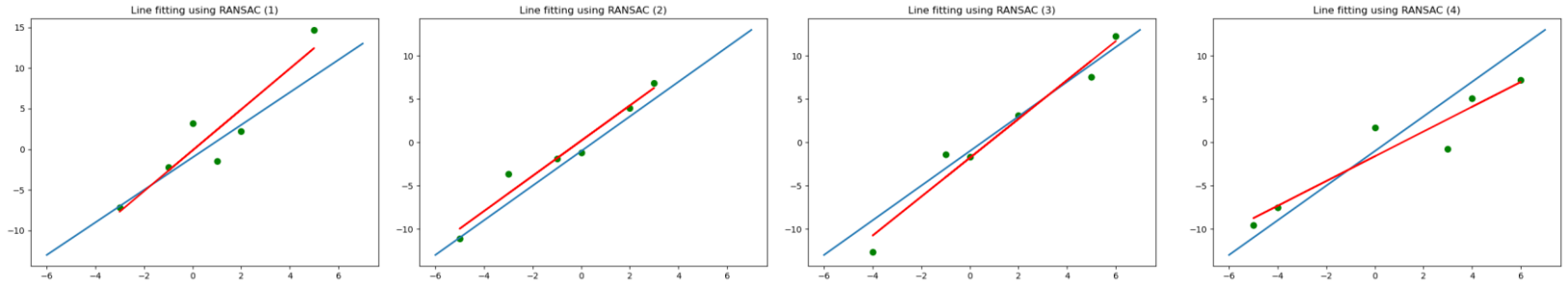
=〉 위 두 방법의 결과를 비교해본다

[Result1]





(iteration 5번 중 error값이 최소인 그래프)



(iteration 5번 중 error가 최소인 그래프 제외 나머지 그래프)

RANSAC criterion : Number of iterations
–〉 5번 돌면 종료

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  10.202460252515817

<When using RANSAC>
Difference between y=2x-1 and fitted line : 5.379803480744247 (graph 5)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

RANSAC을 이용하는 것, 즉 일부 샘플값으로
조건을 만족할 때까지 반복하며 최소 에러를 찾아내서 line fitting하는 방법이
전체 샘플을 사용해서 line fitting하는 것보다 더 나은 결과를 보여준다

[Result2] Result1과 같은 결과가 나오는지 여러 번 테스트 해보았다

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  5.389164889606423

<When using RANSAC>
Difference between y=2x-1 and fitted line : 2.714598475430432 (graph 2)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  5.282908052521204

<When using RANSAC>
Difference between y=2x-1 and fitted line : 3.3828951886658163 (graph 3)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  5.193115408020779

<When using RANSAC>
Difference between y=2x-1 and fitted line : 3.988192573633839 (graph 5)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  5.688880446908234

<When using RANSAC>
Difference between y=2x-1 and fitted line : 0.5241441339562432 (graph 3)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  3.9348756227259347

<When using RANSAC>
Difference between y=2x-1 and fitted line : 3.3845296571462926 (graph 2)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

```
<When using whole samples>
Difference between y=2x-1 and fitted line :  4.4939254210096715

<When using RANSAC>
Difference between y=2x-1 and fitted line : 2.397045911009211 (graph 3)

<Compare the result>
Using RANSAC is closer to y=2x-1 than using whole samples
```

모두 RANSAC을 사용했을 때가 더 좋은 결과를 보이는 것을 볼 수 있다

[Result]

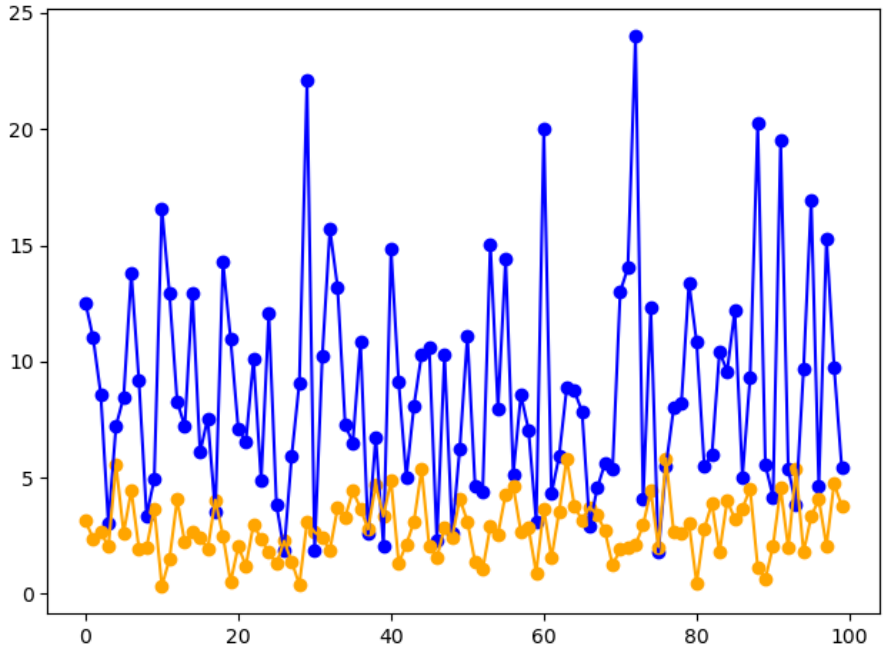
– 100번을 돌려봤는데 대부분의 경우 RANSAC을 사용한 결과가 더 좋았다

[RANSAC] number of iteration : 5

```
100번 중 RANSAC을 사용했을 때 더 좋은 결과 나온 횟수 : 95
100번 중 RANSAC을 사용했을 때 더 안 좋은 결과 나온 횟수 : 5
```

```
100번 중 RANSAC을 사용했을 때 더 좋은 결과 나온 횟수 : 91
100번 중 RANSAC을 사용했을 때 더 안 좋은 결과 나온 횟수 : 9
```

랜덤한 현상을 다루므로
결과는 코드를 돌릴 때마다 다르게 나온다



파란색 부분은 전체 샘플을 사용해서 line fitting 했을 때의 에러 값 분포를 나타내고,

주황색 부분은 RANSAC을 사용해서 line fitting 했을 때의 에러 값 분포를 나타낸다

전체 샘플을 사용했을 때보다 RANSAC을 사용했을 때 더 낮은 오차를 가지고, 편차가 적음을 볼 수 있다

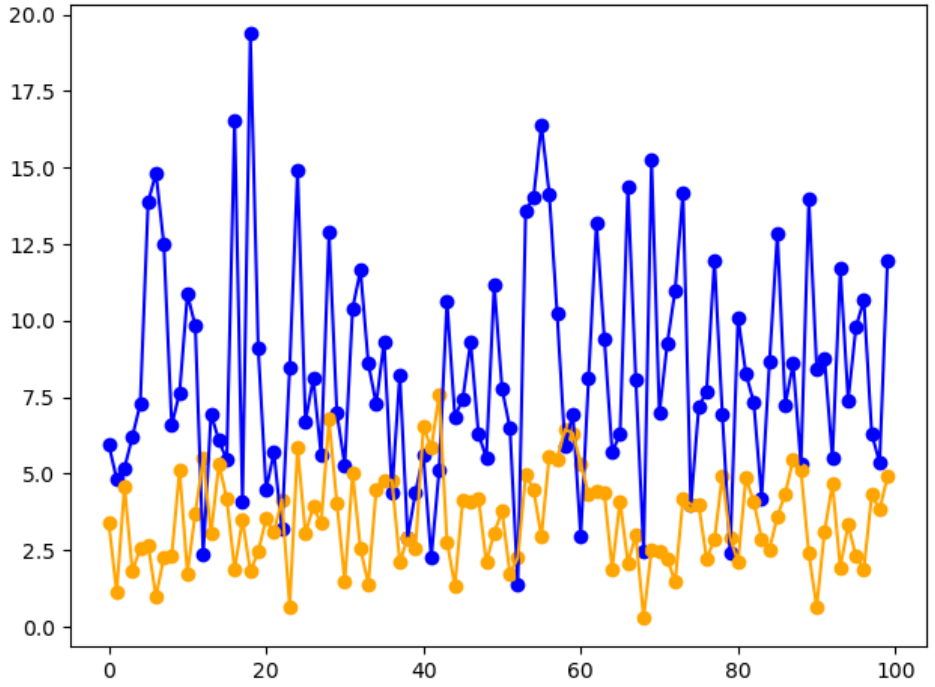
[Result]

– RANSAC의 criterion을 낮추면 대체적으로 RANSAC의 효과를 덜 본다

[RANSAC] number of iteration : 3

```
100번 중 RANSAC을 사용했을 때 더 좋은 결과 나온 횟수 : 87
100번 중 RANSAC을 사용했을 때 더 안 좋은 결과 나온 횟수 : 13
```

```
100번 중 RANSAC을 사용했을 때 더 좋은 결과 나온 횟수 : 89
100번 중 RANSAC을 사용했을 때 더 안 좋은 결과 나온 횟수 : 11
```



앞 분포에 비해 주황색 부분의 오차가 더 들쭉날쭉 하다
RANSAC을 사용했지만 iteration을 적게해서,
최소값을 뽑아도 그 값이 작지 않을 확률이
앞 경우에 비해 크기 때문이다