수치해석 HW#4

RANSAC을 활용하여 line fitting하기

Python 사용

컴퓨터소프트웨어학부 2015005187 최철훈

목차

과정 12개의 점으로 line fitting

1. y = 2x - 1 + N(0, 2)인 노이즈를 더한 식을 만든다.

```
noise = np.random.normal(0, math.sqrt(2), 12)
y = (2 * x - 1) + noise
```

2. 주어진 12개의 점으로 Least Square를 구하여 line fitting 한다.

```
original = np.zeros((12, 2), dtype = np.float32)
for i in range(0, 12):
    original[i, 0] = x[i]
    original[i, 1] = y[i]

X = least_square(original)
print("Original : y = " + str(X[0]) + "x + " + str(X[1]), end=" ")
error = 0
for i in range(0, 12):
    error = error + ((X[0] * original[i, 0] + X[1]) - original[i, 1])**2
print("Error : " + str(error))
```

3. y값들과 line fitting한 방정식과 그에 상응하는 에러값을 구한다.

y값 –

```
[-9.11685231 -7.98844277 -9.18552605 -5.01185655 -2.12129824 -2.01835477
  1.37549009  3.1535108   5.006069    6.75307801  9.61239151 12.70135177]
```

방정식과 에러 -

```
Original : y = [2.025475]x + [-0.7494408] Error : [13.725801]
```

과정 RANSAC으로 line fitting

12개의 점 중 6개의 점을 선택하는 경우는 924번이므로 그 절반인 462번 샘플링하였다.

1. 6개의 점을 랜덤으로 샘플링한다.
2. 랜덤하게 뽑은 6개의 점으로 Least Sqare를 구하여 line fitting한다.
3. 위 과정을 462번 반복하면서 에러값이 가장 작은 line을 구한다.

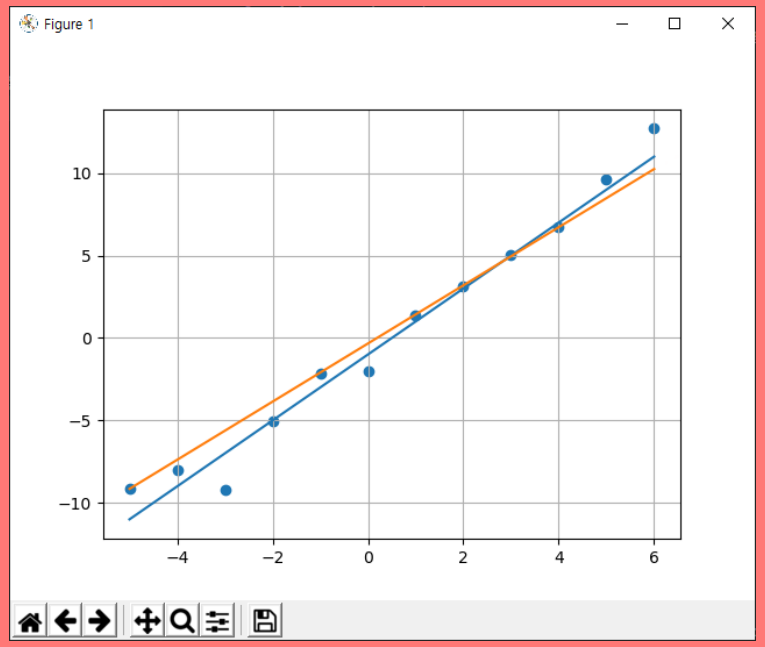
```
sampling = np.zeros((6, 2), dtype = np.float32)
for j in range(0, 462):
    rand = np.random.choice(12, 6, replace=False)

    for i in range(0, 6):
        sampling[i, 0] = x[rand[i]]
        sampling[i, 1] = y[rand[i]]

    X = least_square(sampling)

    error = 0
    for i in range(0, 6):
        error = error + ((X[0] * sampling[i, 0] + X[1]) - sampling[i, 1])**2

    if error < min_error:
        min_error = error
        min_X = X
        min_sample = rand
```

4. 결과로 이전에 구했던 y값에서 4, 5, 7, 8, 10, 11번째의 값으로 line fitting하였을 때가 에러값이 최소가 되었다.

```
[ 4  5  7  8 10 11] y = [1.7634737]x + [-0.3339829] Error : [0.00979441]
```

Compare 비교 결과

앞서 구한 두개의 방정식을 그려보았다. 파란선이 12개의 점으로 fitting한 방정식이고 주황선이 6개의 점으로 fitting한 방정식이다. 육안으로 봤을 때는 무엇이 더 최적의 해인지는 구분할 수 없지만 에러값을 직접 비교하면 샘플링한 직선이 훨씬 최적의 해라는 것을 알 수 있다.



파란 직선 - `Original : y = [2.025475]x + [-0.7494408] Error : [13.725801]`

주황 직선 - `[ 4  5  7  8 10 11] y = [1.7634737]x + [-0.3339829] Error : [0.00979441]`

Compare 여러 번 비교

여러 번 실행해보았다. 결과는 항상 12개의 점으로 fittin했을 때보다 6개의 랜덤 샘플링한 점으로 fitting했을 때의 에러값이 훨씬 작았다.

```
Original : y = [2.0573165]x + [-0.9774945] Error : [17.007288]

[ 2  3  4  5  6 11] y = [2.474002]x + [-0.551824] Error : [0.22234654]
```

```
Original : y = [1.8645313]x + [-1.1623776] Error : [29.898016]

[ 2  3  4  6  8 12] y = [1.9326174]x + [-1.4712017] Error : [0.63589126]
```

```
Original : y = [2.0054486]x + [-1.5574651] Error : [16.884418]

[ 2  4  7  8  9 12] y = [2.0188003]x + [-1.2470587] Error : [0.47639322]
```

```
Original : y = [2.1337123]x + [-1.0153549] Error : [6.827224]

[ 3  5  7  9 10 11] y = [2.4162614]x + [-0.836947] Error : [0.3192285]
```

```
Original : y = [1.9876186]x + [-0.73668665] Error : [28.75656]

[1 2 3 6 8 9] y = [2.0940251]x + [0.58369946] Error : [1.2521672]
```

마무리 느낀점

항상 12개의 점으로 fitting했을 때의 에러보다 RANSAC을 이용한 6개의 점으로 fitting했을 때의 에러가 훨씬 작게 나왔다.
하지만 6개의 점으로 최적의 해를 구한 것이 12개의 점에 대하여도 최적의 해라고 말할 수 있을까?라는 의문이 들었다. 여기서 생각이 든 것이 표본과 모집단의 관계이다. 이번 과제에서는 모집단이 12개밖에 되지를 않지만 만일 모집단이 엄청 크다면 정규분포를 따를 것이고 표본 또한 정규분포를 따르게 되므로 표본이 모집단을 대표한다고 할 수 있을 것이다.

마무리

감사합니다.

https://github.com/cheol-hoon/Numerical_Analysis