수치해석 HW#4

Least Square를 이용하여 2차곡선 fitting하기

Python 사용

컴퓨터소프트웨어학부 2015005187 최철훈

목차

8개의 점을 모두 이용하기 Pseudo-inverse로 구하기

먼저, 주어진 8개의 점을 모두 이용하여 원래의 2차 곡선을 fitting하였다.

1. 8개의 점

```
D = np.array([[-2.9, 35.4], [-2.1, 19.7], [-0.9, 5.7], [1.1, 2.1],
              [0.1, 1.2], [1.9, 8.7], [3.1, 25.7], [4.0, 41.5]])
```

2. Ax=b에서 b행렬 만들기

```
# Make B
B = np.empty((0, 1), dtype=float)
for i in range(0, 8):
    B = np.append(B, [[D[i, 1]]], axis = 0)
```

3. Ax=b에서 A행렬 만들기

```
## Make 1 column of A
A1 = np.empty((0, 1), dtype=float)
for i in range(0, 8):
    A1 = np.append(A1, [[(D[i, 0])**2]], axis = 0)
## Make 2 column of A
A2 = np.empty((0, 1), dtype=float)
for i in range(0, 8):
    A2 = np.append(A2, [[D[i, 0]]], axis = 0)
## Make 3 column of A
A3 = np.ones((8, 1), dtype=float)
## Make A
A = np.hstack([A1, A2])
A = np.hstack([A, A3])
```

4. Pseudo-inverse구하기

```
# pseudo-inverse
At = np.transpose(A)
AtA = np.dot(At, A)
AtAI = np.linalg.inv(AtA)
PI = np.dot(AtAI, At)
```

5. X구하기

```
# X
X = np.dot(PI, B)
print(X)
```

6. 결과

```
[[ 3.16052477]
 [-2.36059821]
 [ 1.35828072]]
```

8개의 점을 모두 이용하기 Fitting함수로 구하기

앞서 구한 X가 맞는지 확인하기 위해 2차곡선을 fitting하는 함수를 이용하여 바로 구해보았다.

1. Fitting함수인 polyfit을 이용하여 X구하기

```
x = np.array([-2.9, -2.1, -0.9, 1.1, 0.1, 1.9, 3.1, 4.0])
y = np.array([35.4, 19.7, 5.7, 2.1, 1.2, 8.7, 25.7, 41.5])

fit = np.polyfit(x, y, 2)
print(fit)
```

2. 결과

```
[ 3.16052477 -2.36059821  1.35828072]
```

앞서 Pseudo-inverse를 이용하여 구한 X와 같은 값이 나오므로 Pseudo-inverse를 이용하여 구하는 과정에서 제대로 코딩하여 결과를 얻었다는 것을 알 수 있다.

6개의 점을 이용하여 fitting하기 과정

주어진 8개의 점 중 6개의 점을 랜덤으로 선택하여 2차 곡선을 fitting하였다.

1. 6개의 점 선택하기

```
# Randomly select 6 points
r = random.sample(range(0, 8), 6)
R = np.empty((0, 2), dtype=float)
for i in r:
    R = np.append(R, [D[i]], axis = 0)
print(R)
```

2. Ax=b에서 b행렬 만들기

```
# Make B
B = np.empty((0, 1), dtype=float)
for i in range(0, 6):
    B = np.append(B, [[R[i, 1]]], axis = 0)
```

3. Ax=b에서 A행렬 만들기

```
## Make 1 column of A
A1 = np.empty((0, 1), dtype=float)
for i in range(0, 6):
    A1 = np.append(A1, [[(R[i, 0])**2]], axis = 0)
## Make 2 column of A
A2 = np.empty((0, 1), dtype=float)
for i in range(0, 6):
    A2 = np.append(A2, [[R[i, 0]]], axis = 0)
## Make 3 column of A
A3 = np.ones((6, 1), dtype=float)
## Make A
A = np.hstack([A1, A2])
A = np.hstack([A, A3])
```

4. Pseudo-inverse구하기

```
# pseudo-inverse
At = np.transpose(A)
AtA = np.dot(At, A)
AtAI = np.linalg.inv(AtA)
PI = np.dot(AtAI, At)
```

5. X구하기

```
# X
X = np.dot(PI, B)
print(X)
```

6개의 점을 이용하여 fitting하기 결과

총 2번 시행하여 2개의 X를 도출하였다.

```
[[-2.1 19.7]
 [ 3.1 25.7]
 [ 1.1  2.1]
 [ 4.  41.5]
 [-0.9  5.7]
 [-2.9 35.4]]
[[ 3.18054193]
 [-2.39915013]
 [ 1.16085405]]
```

첫 번째 6개의 선택된 점과 그에 따른 결과

```
[[ 4.  41.5]
 [ 1.9  8.7]
 [ 1.1  2.1]
 [-2.1 19.7]
 [ 0.1  1.2]
 [-0.9  5.7]]
[[ 3.07895662]
 [-2.24365274]
 [ 1.3153594 ]]
```

두 번째 6개의 선택된 점과 그에 따른 결과

Compare 두 곡선의 비교

앞서 6개의 점으로 fitting한 두 결과에 대한 2차 곡선 그래프를 그려 비교해보았다.

```
[[-2.1 19.7]
 [ 3.1 25.7]
 [ 1.1  2.1]
 [ 4.  41.5]
 [-0.9  5.7]
 [-2.9 35.4]]
[[ 3.18054193]
 [-2.39915013]
 [ 1.16085405]]
```

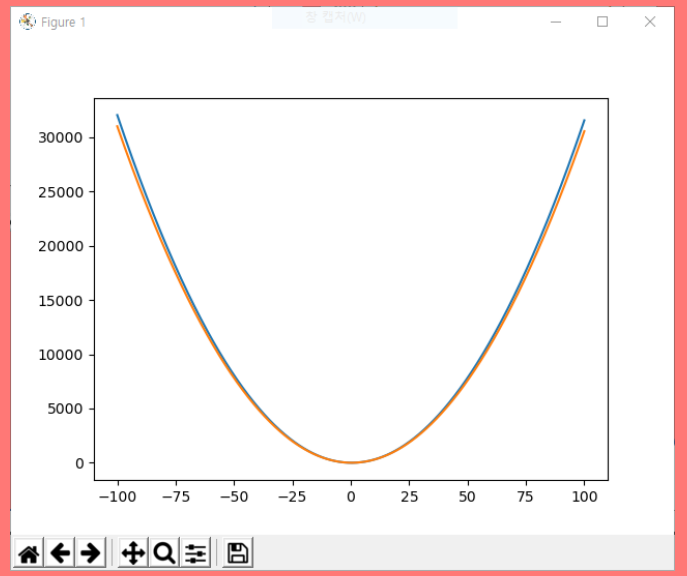
파랑곡선

```
[[ 4.  41.5]
 [ 1.9  8.7]
 [ 1.1  2.1]
 [-2.1 19.7]
 [ 0.1  1.2]
 [-0.9  5.7]]
[[ 3.07895662]
 [-2.24365274]
 [ 1.3153594 ]]
```

주황곡선

```
    if j == 1:
        z1 = X[0, 0]
        y1 = X[1, 0]
        x1 = X[2, 0]
    elif j == 2:
        z2 = X[0, 0]
        y2 = X[1, 0]
        x2 = X[2, 0]

k = np.arange(-100, 100, 0.01)
fit1 = z1*k**2 + y1*k + x1
plt.plot(k, fit1)
fit2 = z2*k**2 + y2*k + x2
plt.plot(k, fit2)
plt.show()
```



두 곡선의 비교 결과 큰 차이가 없음을 알 수 있다. 그나마 큰 차이가 있다면 극솟값에서 멀어질수록 조금씩 벌어진다는 것이다. 이외에도 여러 번 시도해 보았으나 매번 비슷하게 큰 차이가 없었지만 간혹 극솟값에서 멀어질수록 벌어지는 경우가 나왔다.

Compare 원래 곡선과의 비교

8개의 fitting한 그래프와 6개의 점으로fitting한 그래프를 비교해보았다.

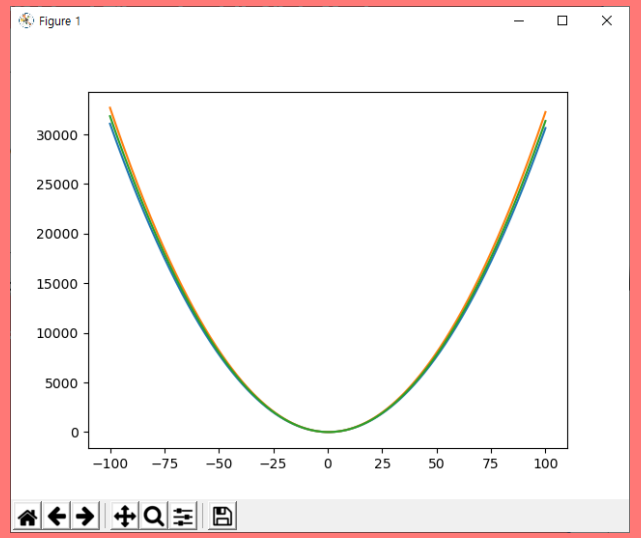
```
[[ 4.   41.5]
 [ 1.1   2.1]
 [ 0.1   1.2]
 [ 3.1  25.7]
 [ 1.9   8.7]
 [-2.1  19.7]]
[[ 3.086668  ]
 [-2.14233456]
 [ 1.44708834]]
```

파랑곡선

```
[[ 0.1   1.2]
 [ 1.1   2.1]
 [ 1.9   8.7]
 [-0.9   5.7]
 [ 3.1  25.7]
 [-2.1  19.7]]
[[ 3.24789136]
 [-2.1379364 ]
 [ 1.01464205]]
```

주황곡선

```
k = np.arange(-100, 100, 0.01)
fit1 = z1*k**2 + y1*k + x1
plt.plot(k, fit1)
fit2 = z2*k**2 + y2*k + x2
plt.plot(k, fit2)
fit3 = 3.16052477*k**2 - 2.36059821*k + 1.35828072
plt.plot(k, fit3)
plt.show()
```



초록색이 8개의 점으로 fitting한 원래의 2차 곡선이다. 두 곡선과 원래 곡선의 비교 결과 앞선 6개의 점으로 fitting한 두 곡선과의 차이와 비슷하게 큰 차이가 없음을 알 수 있다. 그나마 큰 차이가 있다면 극솟값에서 멀어질수록 조금씩 벌어진다는 것이다. 이외에도 여러 번 시도해 보았으나 매번 비슷하게 큰 차이가 없었다. 이를 통해 점의 개수가 많을수록 더 정확한 곡선을 fitting할 수 있다는 사실을 알 수 있다.

마무리

감사합니다.

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