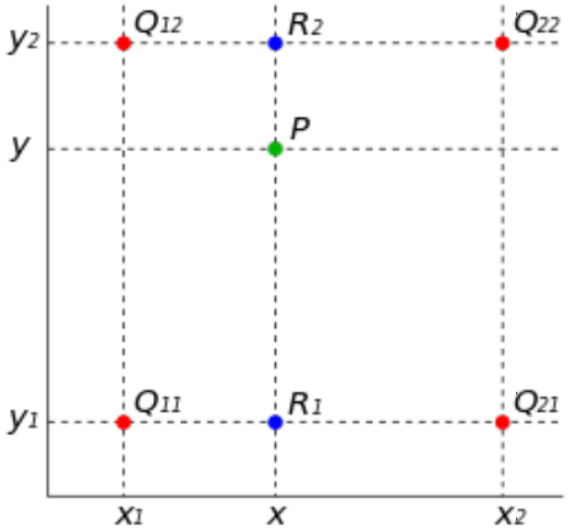
Interpolation and Sampling

2017135002/최성윤

기본원리(Two dimension Interpolation)

1)Nearest Neighbor



1차원 경우와 유사하게 가장 가까이 있는 data를 사용한다
위의 사진의 경우 P는 Q_{12}의 값을 가진다

2)Bilinear Interpolation

2차원 인 경우 1차원 interpolation을 2번 수행
Ex)x축에 대하여 interpolation -> y축에 대하여 interpolation
1.X축에 대하여 interpolation

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

2.y축에 대하여 ineterpolation

$$\begin{aligned} f(x,y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &\approx \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \left(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1) \right) \end{aligned}$$

$$f(x, y) = Af(Q_{11}) + Bf(Q_{21}) + Cf(Q_{12}) + Df(Q_{22})$$

3)Bicubic Interpolation

Bilinear Interpolation와 유사하게
Cubic Interpolation을 2번 실행한다.

실행

1.Nearest Neighborhood

```
data1=fits.open('image1.fits')
image1=data1[0].data
```

Image를 가져올 data를 불러온다

```
nearest=np.zeros((500,500))
for i in range(500):
    for j in range(500):
        nearest[int(i)][int(j)]=image1[int(i/5)][int(j/5)]
```

500x500을 가지는 array 생성후 , 각각 좌표에 image1에 대하여 nearest neighborhood 실행

```
plt.imshow(nearest,cmap='gray')
```

2.Bilinear Interpolation

```
a=np.linspace(0,100,100)
b=np.linspace(0,100,100)
x=np.linspace(0,100,500)
y=np.linspace(0,100,500)
```

Scipy.interpolation을 사용하여 bilinear interpolation 진행

```
bilnear1=interpolate.interp2d(a,b,image1,kind='linear')
bilnear2=bilnear1(x,y)
```

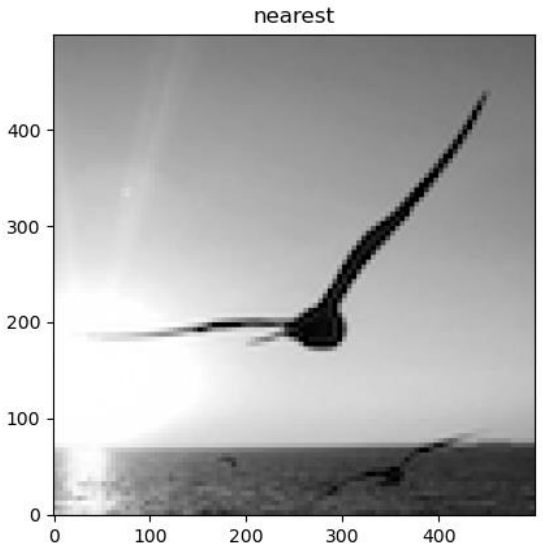
실행

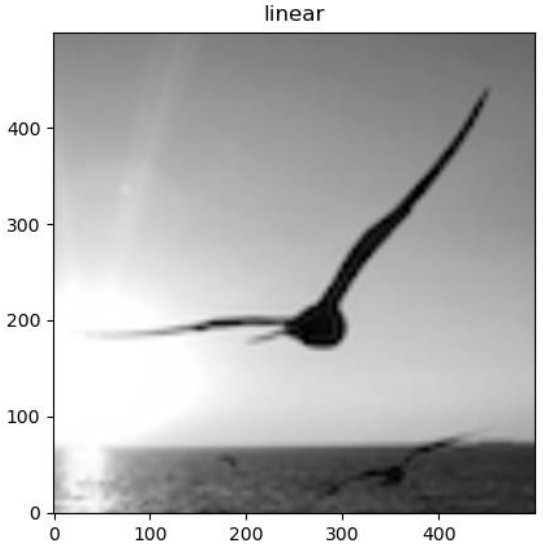
3.Bicubic Interpolation

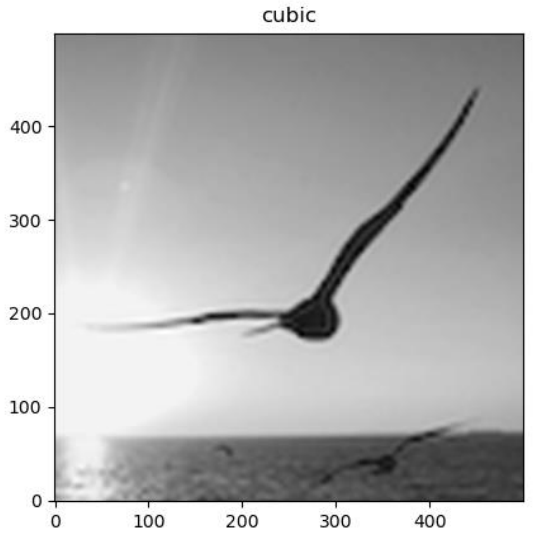
```
cubic1=interpolate.interp2d(a,b,image1,kind='cubic')
cubic2=cubic1(x,y)
```

Scipy.interpolation을 사용하여
bicubic interpolation 진행

결과



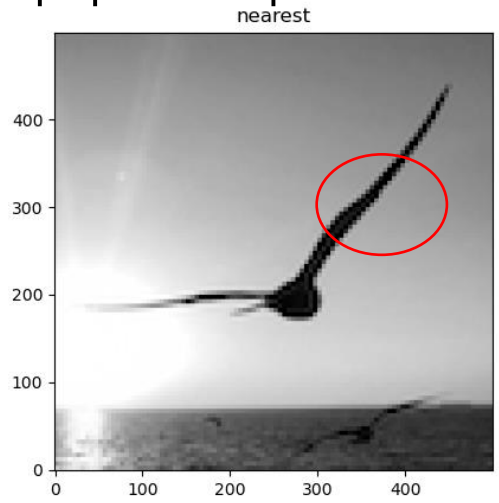


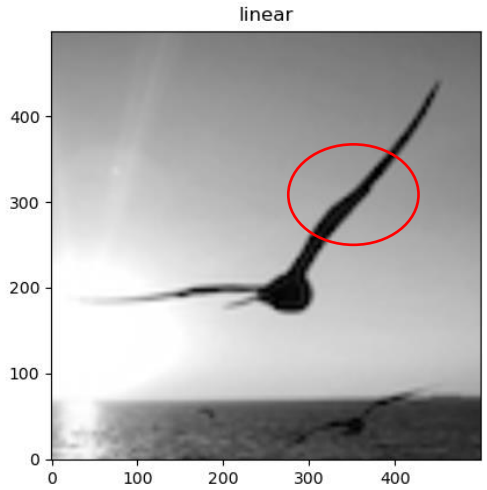


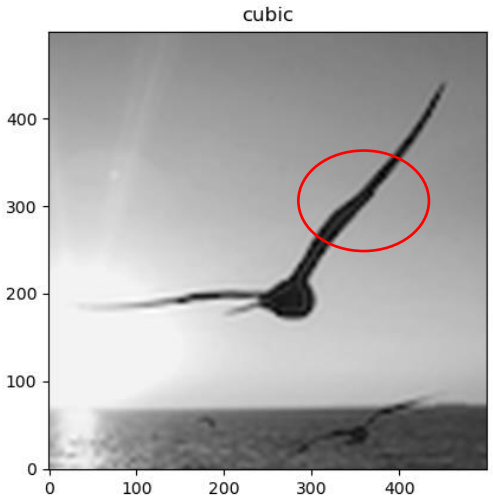
토의사항

3가지 Interpolation 비교

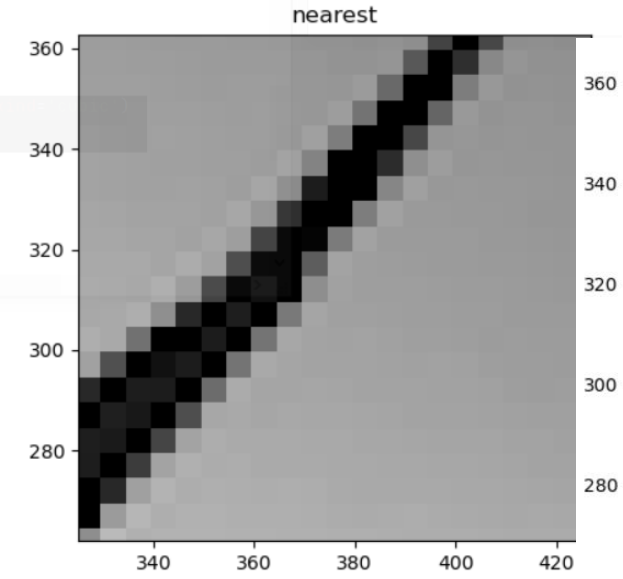
1.시각적으로 비교

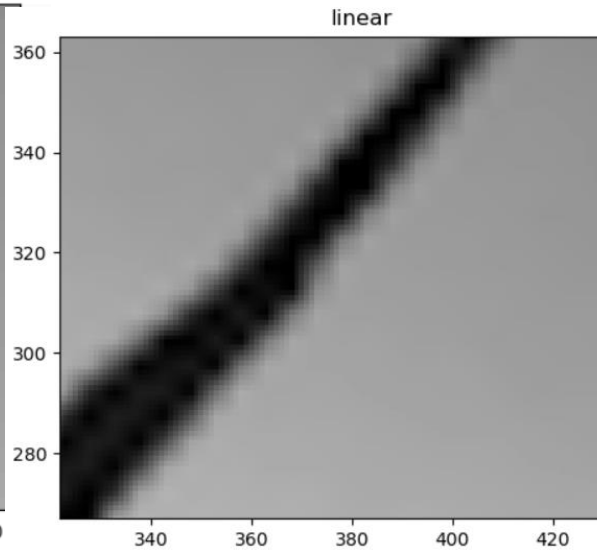


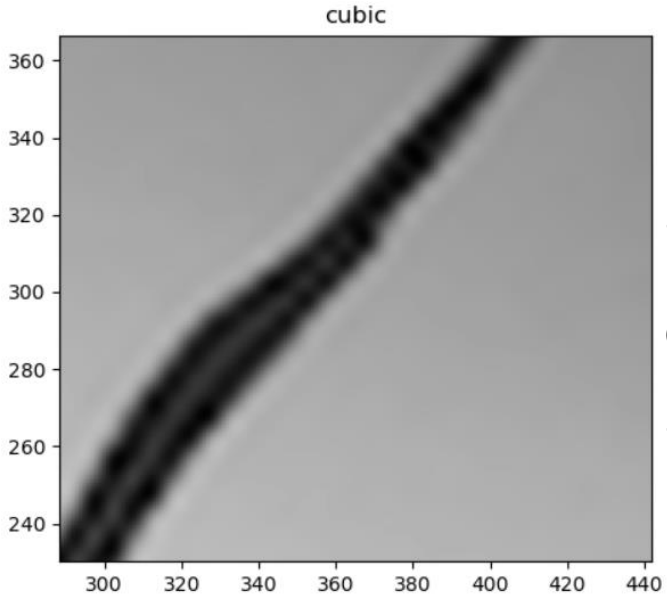




Nearest 경우 눈에 띄게 화질이 좋지 않고 계단현상이 나타난다.
원 부분을 확대해서 본다면?





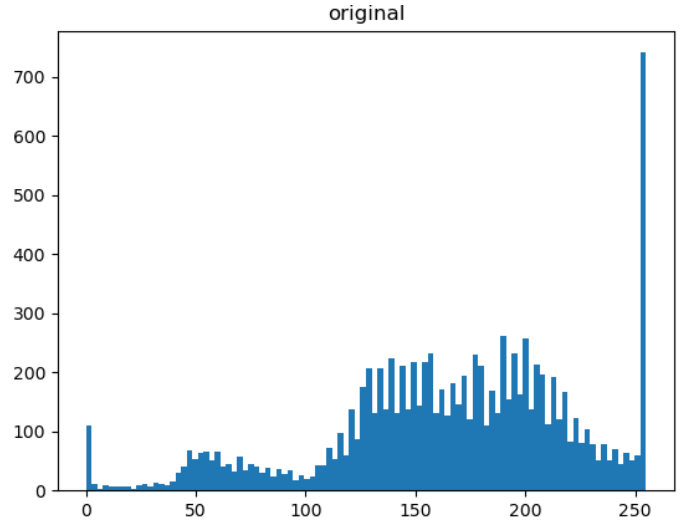


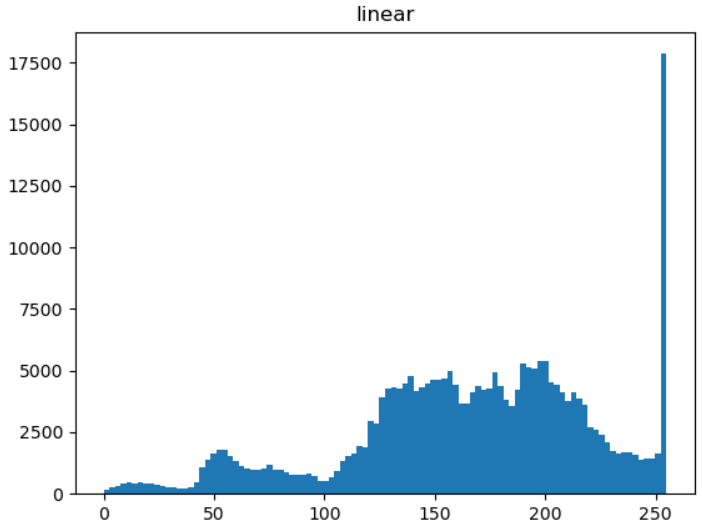
Nearest의 경우에는 확실히 부드럽지 못하고 계단현상이 크게 나타남을 볼 수 있다.

Linear의 경우에는 Nearest보다 선명하고 계단현상이 많이 줄어들었다

Cubic의 경우에는 나머지 2개의 비하여 선명하고 계단현상도 많이 없음을 볼수있다.

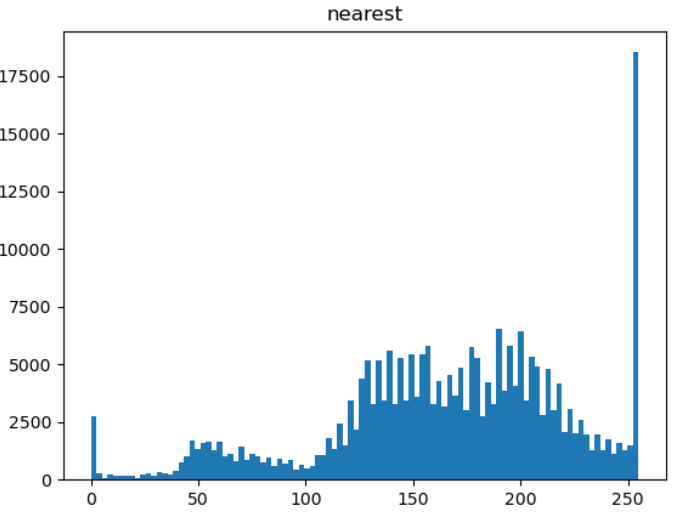
2.Data 히스토그램을 통하여 비교

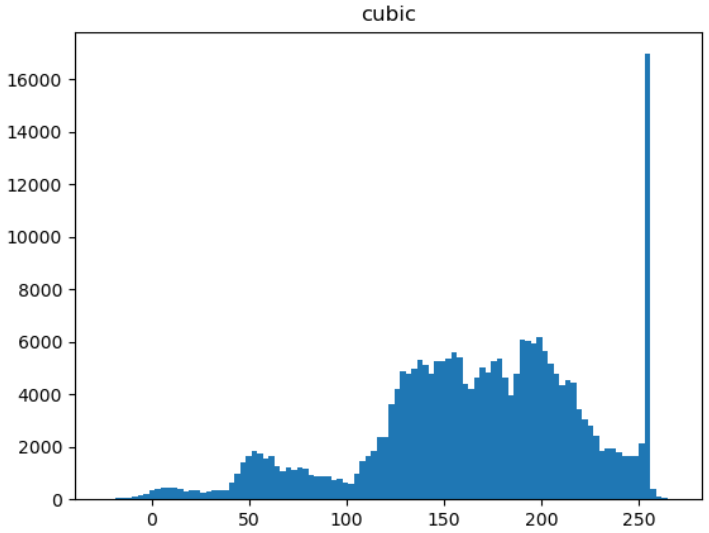




각각 original, nearest, bilinear, bicubic의 경우에서의 data 값들의 분포를 히스토그램으로 나타내었다

Nearest의 경우 original image값들을 그대로 사용하기때문에 data들의 분포가 original과 동일함을 볼수 있다.
분포가 고르지 않고 어느 특정 pixel 부분만 많음을 알수있다.
->pixel data들이 부드럽지 못하다





Linear와 cubic은 비슷한 히스토그램을 가진다. 둘 다 nearest의 분포보다 더 부드럽게 이어짐을 확인할수있다.

하지만 그중에서도 linear가 조금 더 부드러움을 확인할수 있다.
->pixel data들이 부드럽게 이어진다.

3.PSNR을 통하여 비교

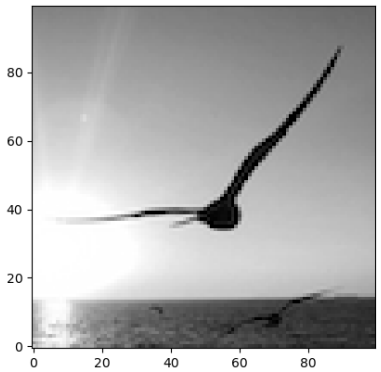
PSNR은 peka signal-to-noise-ratio를 의미한다.
이는 원본이미지와 압축이미지 사이의 품질을 측정할때 사용된다.

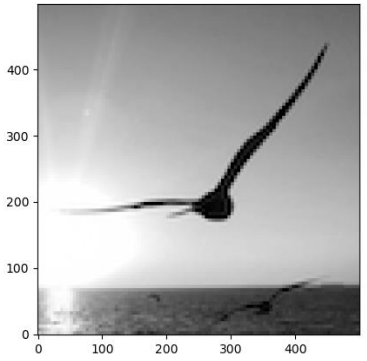
$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

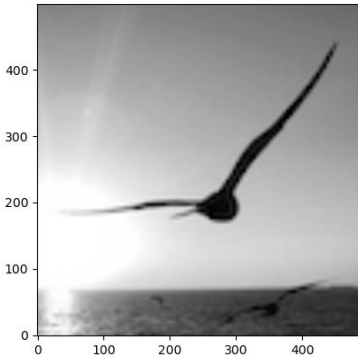
MSE는 평균제곱오차를 나타내는데
원본이미지와 비교하는 이미지의 데이터 누적 제곱 오류를 나타낸다.

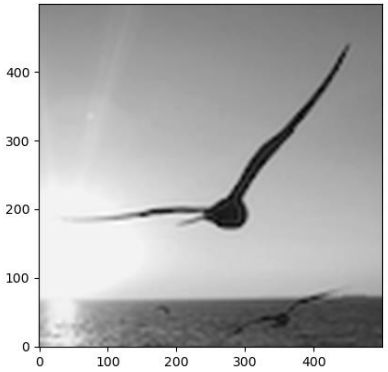
$$PSNR = 20 \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right)$$

PSNR은 이 MSE의 오류정도 척도를 나타태난다.
흔히 원본이미지와 압축이미지를 비교할때
MSE가 낮을수록 (PSNR은 클수록) 오류가 낮다고 한다.









각각 original , nearest , bilinear, bicubic 이미지를 plt.savefig()를 통하여 저장후
이를 imread을 사용하여 픽셀 data값을 읽어낸다. (640x480의 크기를 가진다)
**100x100, 500x500의 데이터를 가지는 이미지를 모두 다 640x480로 저장하기때문에 오차가 커질것이다

```python
def psnr(img1,img2):
    mse = np.mean((img1 - img2) ** 2)
    print("mse:",mse)
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * ma.log10(PIXEL_MAX / ma.sqrt(mse))
d = psnr(original, contrast)
print(d)
```

PSNR을 통하여 각각 original 사진과
Nearest, bilinear, bicubic에 대하여 비교 하였다.

Nearest

```
mse: 1.938662109375
45.25578238649048
```

Bilinear

```
mse: 5.565826822916667
40.67550671450146
```

Bicubic

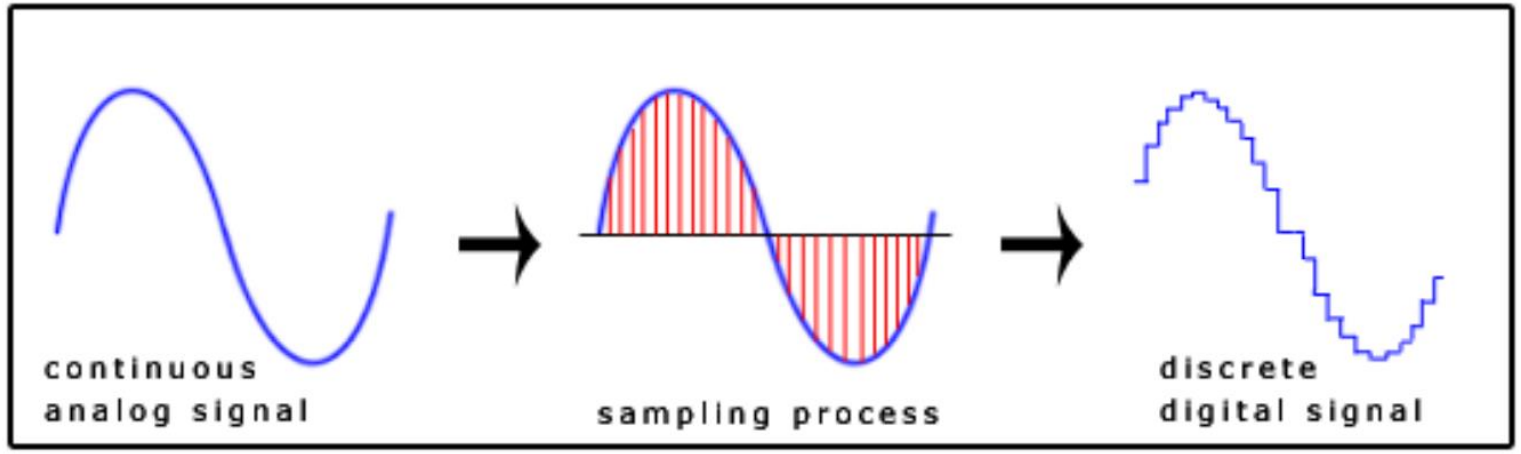
```
mse: 20.040478515625
35.11172273701926
```

PSNR을 확인하였을때는 Nearest가 더 품질이 좋은걸로 나온다. 왜그럴까?

PSNR은 기본적으로 원본 이미지와의 오차를 나타낸다.
하지만 Nearest의 원리는 가장 가까운 원본이미지의 data값 그대로 사용하고
Bilinear, Bicubic의 경우는 원본데이터값들을 interpolation을 사용하기 때문에 원본 데이터와는 차이가 난다.
따라서 이경우에는 Bicubic으로 갈수록 PSNR이 작게 나옴을 알수있는데
이를 반대로 해석하면 Bicubic일수록 다양한 pixel data를 사용하여 원본이미지를 확대 하였기때문에 PSNR이 작게 나오는걸로 해석해볼수 있다. ->다양한 pixel data->이미지를 더 부드럽고 선명하게

기본원리

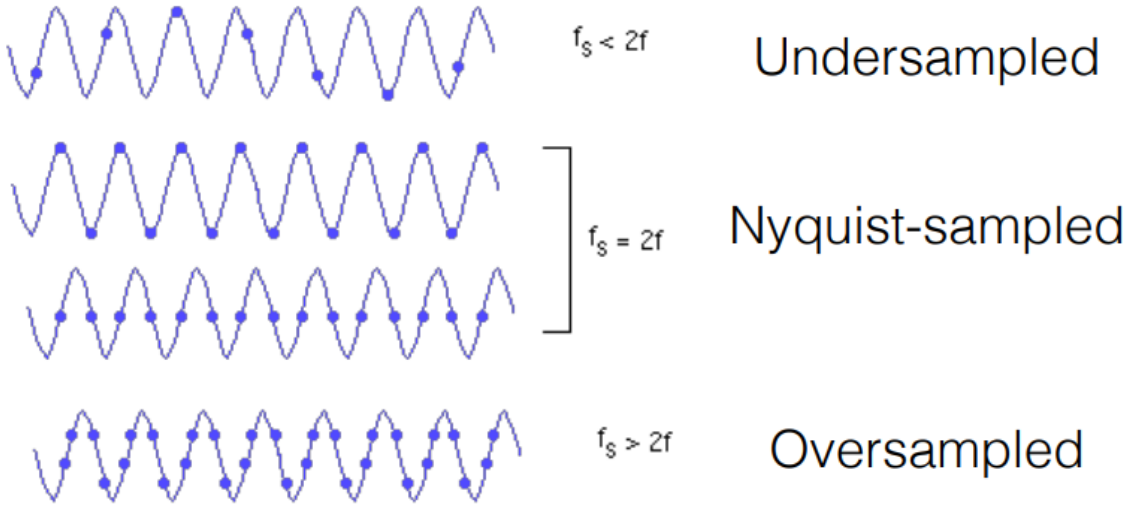
Sampling



Sampling은
일부분의 data를 사용하여
함수를 추정하는 방법이다

Nyquist Sampling Theorem

2 sampling points per wavelength



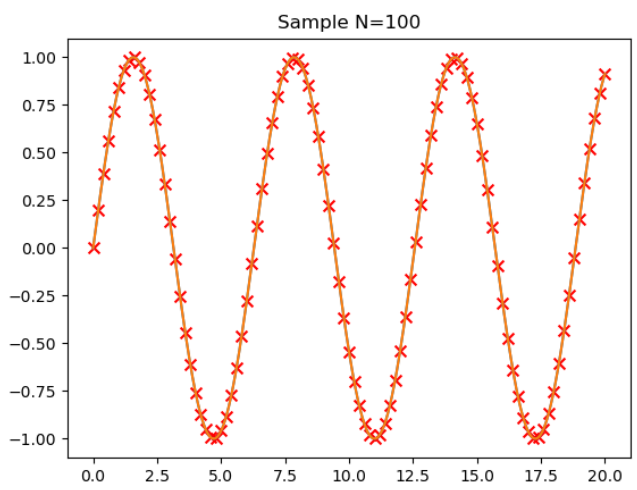
Sampling 하는 데이터의 주파수는 2Q Hz이어야한다.
Q는 F(x)의 최대 주파수
(한 파장당 2개의 data)

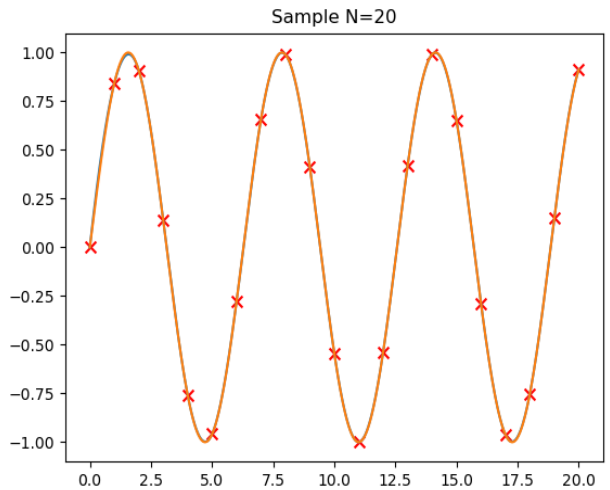
실행

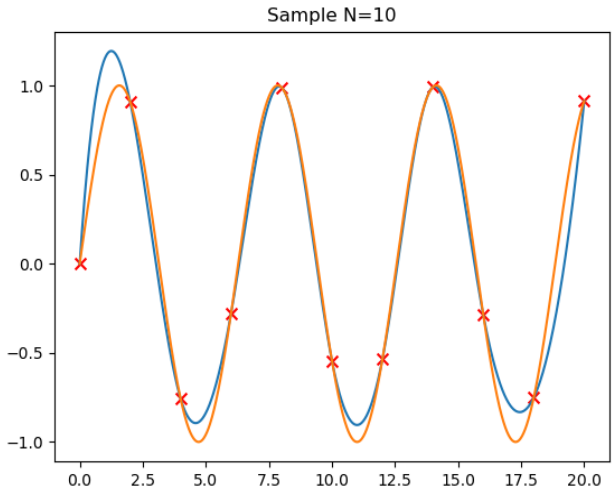
```
t1=np.linspace(0,20,101)
t2=np.linspace(0,20,1000)
x=np.sin(t1)
cb=cs(t1,x)
plt.scatter(t1,x,s=50,c='r',marker='x')
plt.plot(t2,cb(t2),t2,np.sin(t2))
plt.title('Sample N=100')
plt.show()
```

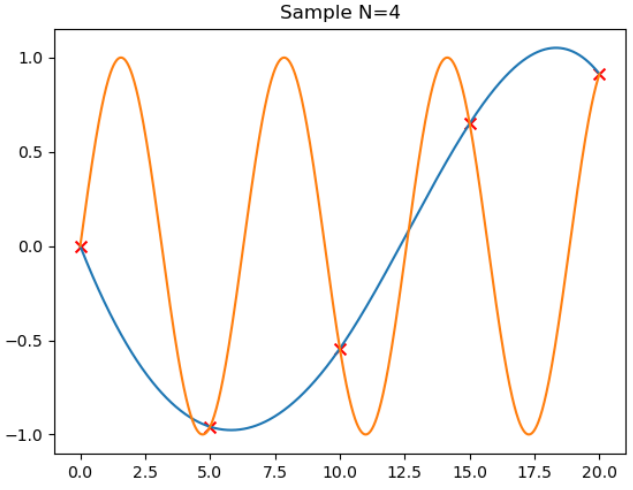
0~20까지를 100,20,10,4개 의 균등한 간격으로 해당지점을 sample 한 뒤 이를 바탕으로 cubic spline을 실시하여 결과값과 이론값을 비교해 보았다.

결과









토의사항

결과의 사진에서 초록색 그래프가 sampling 한 data로 interpolation 한 함수, 빨간색 그래프는 sin(x) 그래프이다.
N=100,20인 경우는 원래의 그래프와 거의 일치함을 보이며 N=10 인경우는 약간의 오차가 있지만 비슷함을 보인다.
하지만 N=4인 경우는 완전 다른 그래프를 보인다. (N-1은 data 개수)

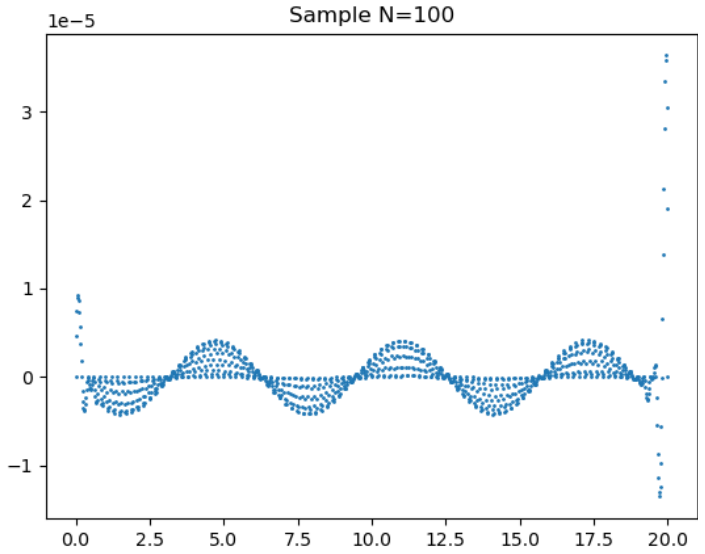
위 경우의 Sin(x)의 Nyquist Sampling Theorem는 3.25만큼의 파장이 있으므로 2를 곱한다면 6.5로
최소 7개의 data가 있어야한다.
따라서 N=10부터는 어느정도 SIN 그래프를 따라가지만 N=4(data 개수는 5개)인경우는 완전히 다른 그래
프 양상을 보여준다

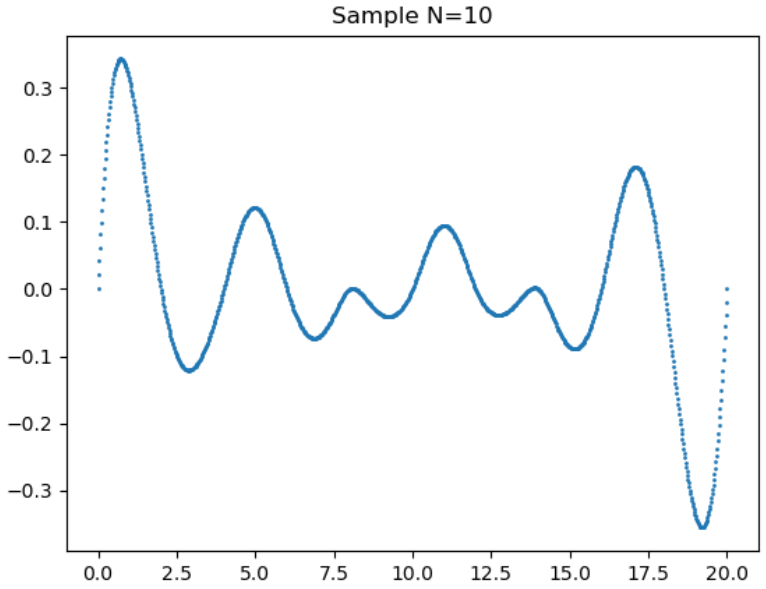
N=10,20,10 인 경우는 Oversampled
N=4 인 경우는 Undersampled

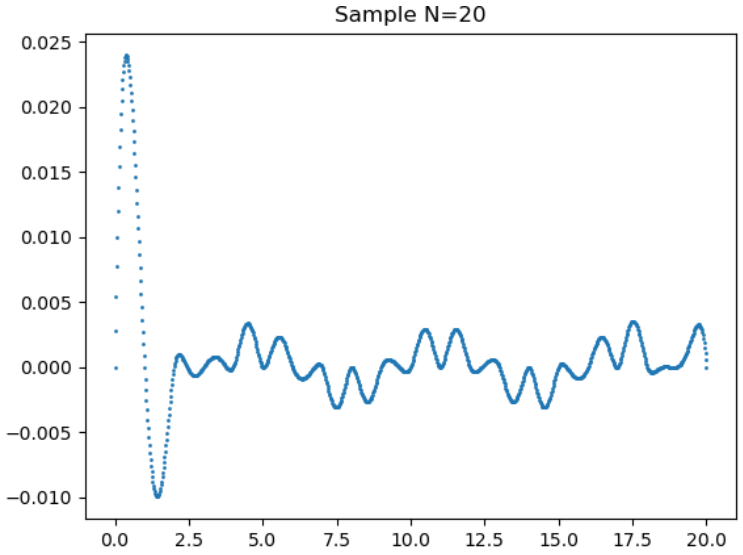
Nyquist Sampling Theorem 보다 적은 개수를 가지는 sample로 함수를 interpolating 하였을때
Aliasing 발생

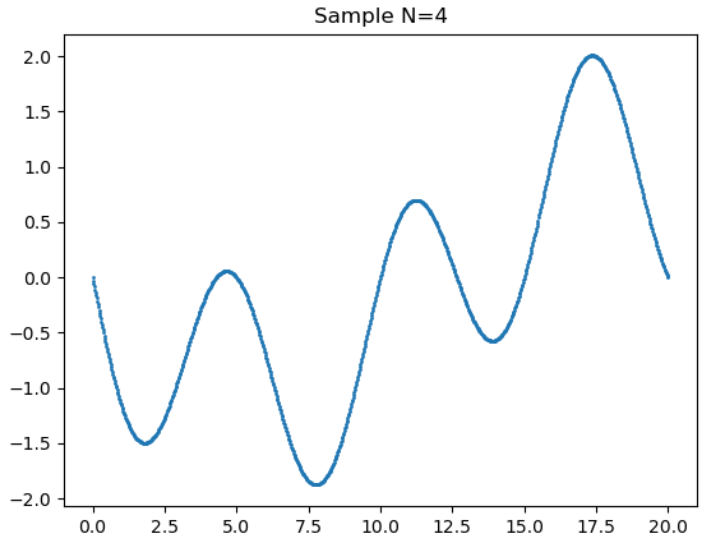
토의사항

원래 함수와 오차









각각 sample 한 함수값들을
원래 값(sinx)와의 오차를 구해보았다.

n=100 인경우 오차가 매우 작음을
확인할수 있다

N=20인 경우 오차가 있기는 하지만 의미있는
정도는 아님을 알 수 있다.

N=10인 경우 어느 부분에서는 오차가 꽤 크지
만 대체로 작음을 알 수있다.

N=4 인경우 대부분 경우에서 오차가 큼을 볼
수 있다.

Reference

Peak signal-to-noise ratio,Wikipedia

https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

V. Diana Earshia and M. Sumathi ,A Comprehensive Study of 1D and 2D Image Interpolation Techniques

Sudip Kumar and Neelesh Agrawal, Performance Analysis Of Different Interpolation Technique Used For Improving PSNR Of Different Images Using Wavelet Transform

Peak signal-to-noise ratio(신호대 잡음비),2019/01/01

https://blog.naver.com/chrhdhkd/221431763390