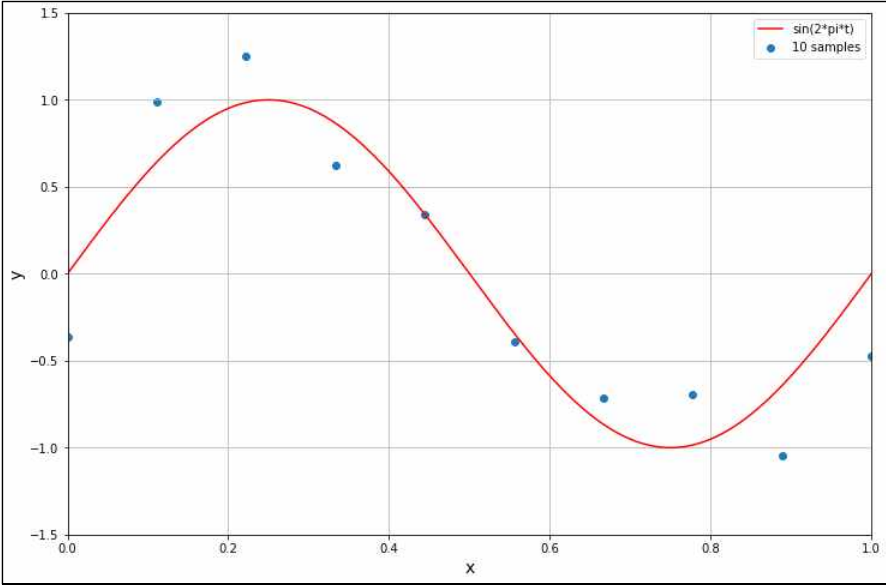


| | | | | | |
|-----|-----|----|-----------|----|-----|
| 과제명 | HW1 | 학번 | 201611182 | 이름 | 최동현 |
|-----|-----|----|-----------|----|-----|

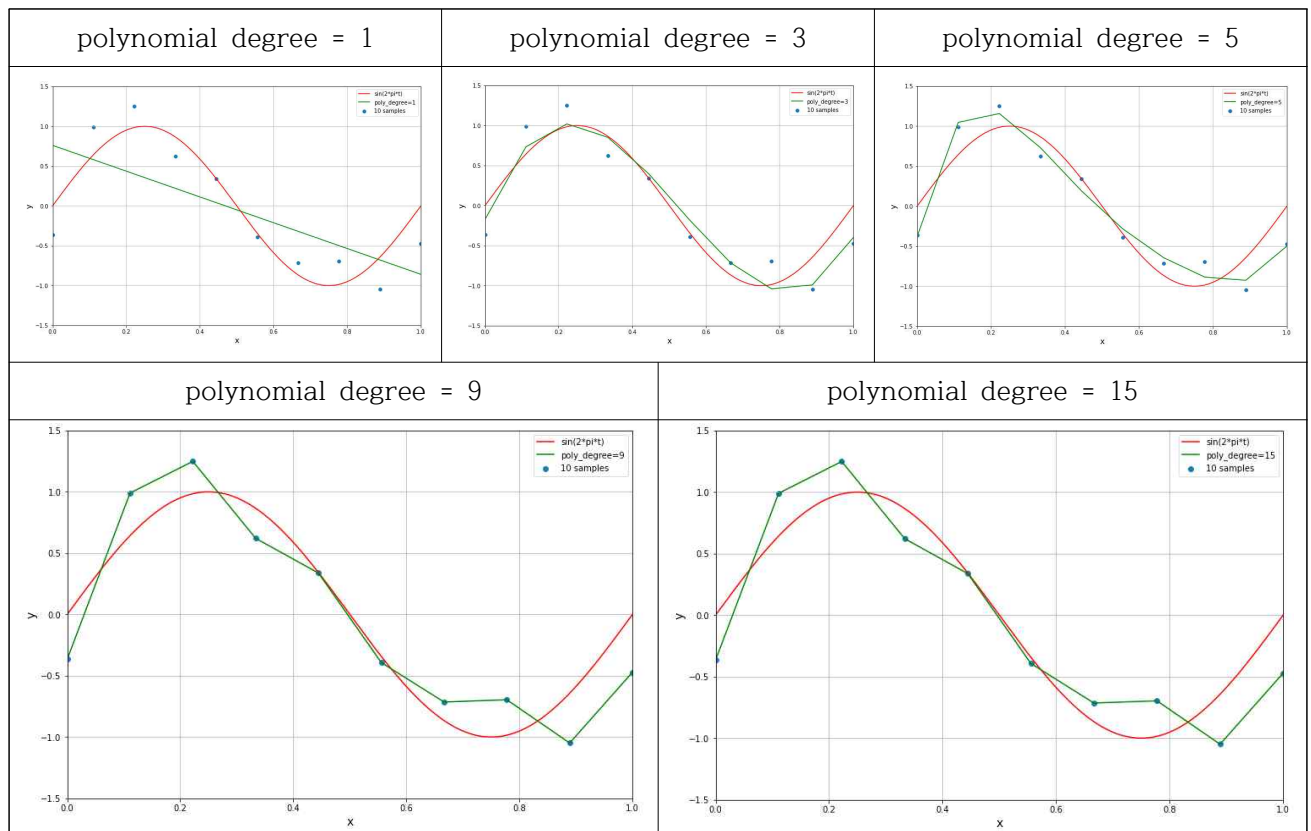
| | |
|--------|---|
| 보고서 내용 | 주요코드, 결과, 결과에 대한 분석을 기술한 보고서(python colab 활용) |
|--------|---|

| 문제 번호 | 내용 |
|-------|---|
| | <div>○ 주요 코드</div> <pre> 1 %matplotlib inline 2 import numpy as np 3 import matplotlib.pyplot as plt 4 import random 5 rcParams['figure.figsize'] = 12, 8 6 7 random.seed(1) 8 datalen = 10 9 pi = np.pi 10 11 def sin(x): # sin(x) 12 return np.sin(x) 13 14 def cos(x): # cos(x) 15 return np.cos(x) 16 17 x = np.linspace(0, 1, 1000) 18 y = sin(2*pi*x) 19 x_s = np.linspace(0, 1, 10) 20 #y_s = sin(2*pi*x_s)+random.uniform(-0.5, 0.5) 21 y_s = [sin(2*pi*(k))+random.uniform(-0.5, 0.5) for k in x_s] 22 23 plt.xlim(0,1) 24 plt.ylim(-1.5,1.5) 25 26 plt.plot(x, y, color = 'r', label = 'sin(2*pi*t)') 27 plt.scatter(x_s, y_s, label = '10 samples') 28 plt.legend() 29 30 plt.xlabel("x", size=14) 31 plt.ylabel("y", size=14) 32 plt.grid() 33 34 plt.show() </pre> <div>1번 ○ 분석 결과</div>  <div> <ul style="list-style-type: none"> - $\sin(2\pi x)$함수를 기반으로 Gaussian noise로 random.uniform 노이즈 -0.5~0.5를 부여해 plot한 결과 - 10개의 samples를 plot한 결과 </div> |

○ 주요 코드

```
1 poly_features = PolynomialFeatures(degree=1, include_bias=False)
2 x_poly = poly_features.fit_transform(x_s.reshape(-1,1))
3
4
5 lr = LinearRegression().fit(x_poly, y_s)
6
7 plt.plot(x, y, color = 'r', label = 'sin(2*pi*t)')
8 plt.plot(x_s, lr.predict(x_poly), color = 'g', label = 'poly_degree=1')
9 plt.scatter(x_s, y_s, label = '10 samples')
10
11 plt.legend()
12 plt.xlim(0,1)
13 plt.ylim(-1.5,1.5)
14 plt.xlabel("x", size=14)
15 plt.ylabel("y", size=14)
16 plt.grid()
17
18 plt.show()
```

○ 분석 결과



- 1, 3, 5, 9, and 15 degree에 따라 형성된 polynomial basis function을 만들고, regression lines를 그린 결과이다.
- samples의 개수가 적은 만큼 polynomial degree = 9에서 모든 samples를 예측하는 regression line이 형성됨을 확인할 수 있다.

○ 주요 코드

```

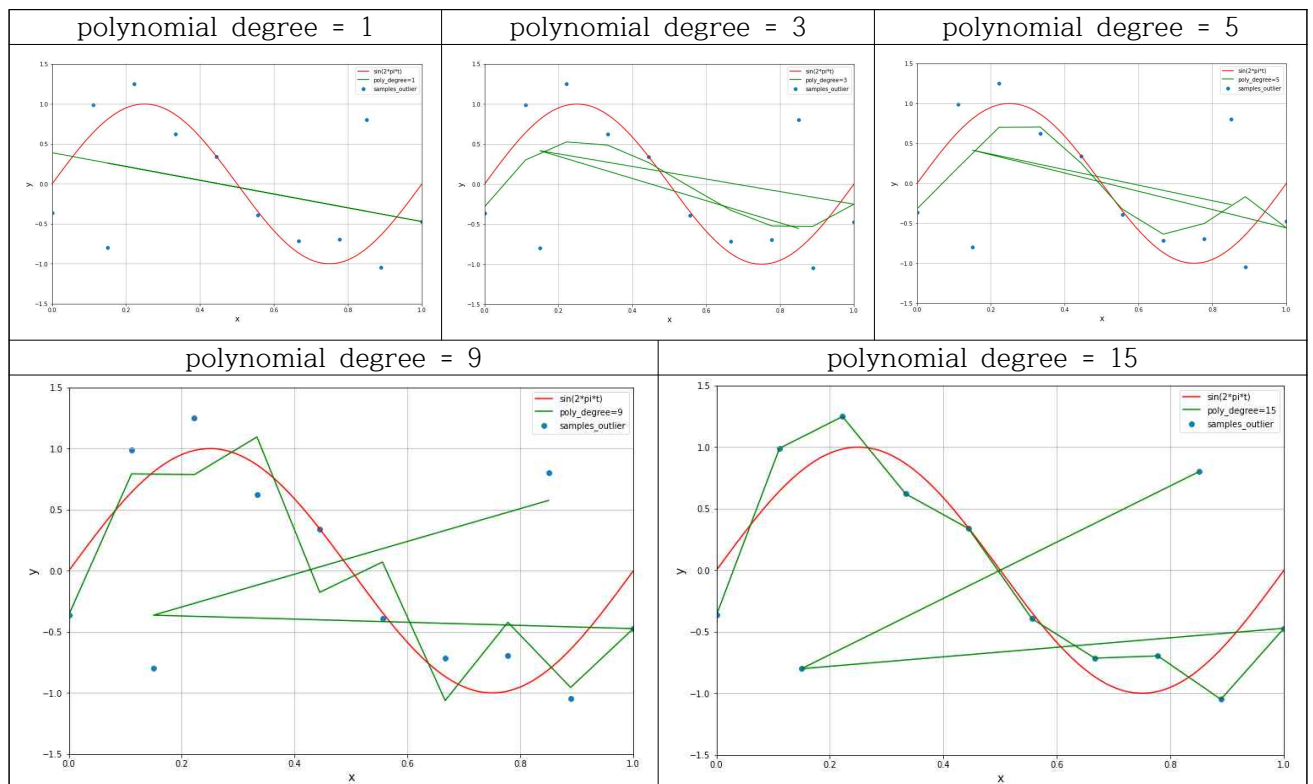
1 x_s_e = np.append(x_s, [0.15, 0.85])
2 x_s_e
3 y_s_e = np.append(y_s, [-0.8, 0.8])
4 y_s_e
5
6 poly_features = PolynomialFeatures(degree=1, include_bias=False)
7 x_poly = poly_features.fit_transform(x_s_e.reshape(-1,1))
8
9
10 lr = LinearRegression().fit(x_poly, y_s_e)
11
12 plt.plot(x, y, color = 'r', label = 'sin(2*pi*t)')
13 plt.plot(x_s_e, lr.predict(x_poly), color = 'g', label = 'poly_degree=1')
14 plt.scatter(x_s_e, y_s_e, label = 'samples_outlier')
15
16 plt.legend()
17 plt.xlim(0,1)
18 plt.ylim(-1.5,1.5)
19 plt.xlabel("x", size=14)
20 plt.ylabel("y", size=14)
21 plt.grid()
22
23 plt.show()

```

- x_s_e와 y_s_e의 변수에 아웃라이어 (0.15, -0.8)과 (0.85, 0.8) 2개를 추가했다.

○ 분석 결과

3번



- sin함수를 따르지 않는 outlier를 2개 추가하여 1, 3, 5, 9, and 15 degree에 따라 형성된 polynomial basis function을 만들고, regression lines를 그린 결과이다.
- outlier가 없을 때와 다르게, polynomial degree = 9에서 예측 정확도가 떨어짐을 직관적으로 확인할

수 있었다.

- 실패요인이 있다면, regression line이 겹치는 지점이 있다는 것이다. sample 데이터를 sequential 형태로 학습 시켜서 생겨난 결과인 것으로 분석하였다.
- polynomial degree = 15에서 모든 samples를 예측하는 regression line이 생성된 것을 확인할 수 있었다. 차수가 sample데이터 개수보다 많아서 overfitting된 결과임을 확인하였다.

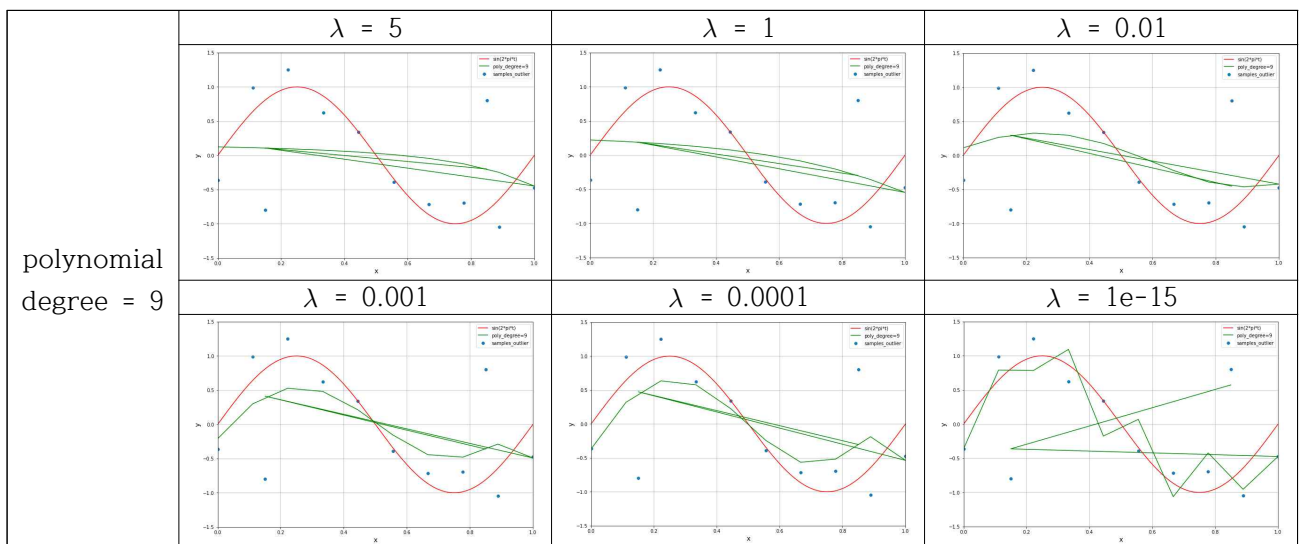
○ 주요 코드 <L2(Ridge) 규제>

```
1 #L2 규제(Ridge)
2 from sklearn.linear_model import Ridge
3 rcParams['figure.figsize'] = 12,8
4
5 poly_features = PolynomialFeatures(degree=15, include_bias=False)
6 x_poly = poly_features.fit_transform(x_s_e.reshape(-1,1))
7
8
9 ridge = Ridge(alpha=0.0001).fit(x_poly, y_s_e)
10
11 plt.plot(x, y, color = 'r', label = 'sin(2*pi*x)')
12 plt.plot(x_s_e,ridge.predict(x_poly),color = 'g', label = 'poly_degree=15')
13 plt.scatter(x_s_e, y_s_e, label = 'samples_outlier')
14 plt.legend()
15
16 plt.xlim(0,1)
17 plt.ylim(-1.5,1.5)
18 plt.xlabel("x", size=14)
19 plt.ylabel("y", size=14)
20 plt.grid()
21
22 plt.show()
```

○ 분석 결과 <L2(Ridge) 규제>

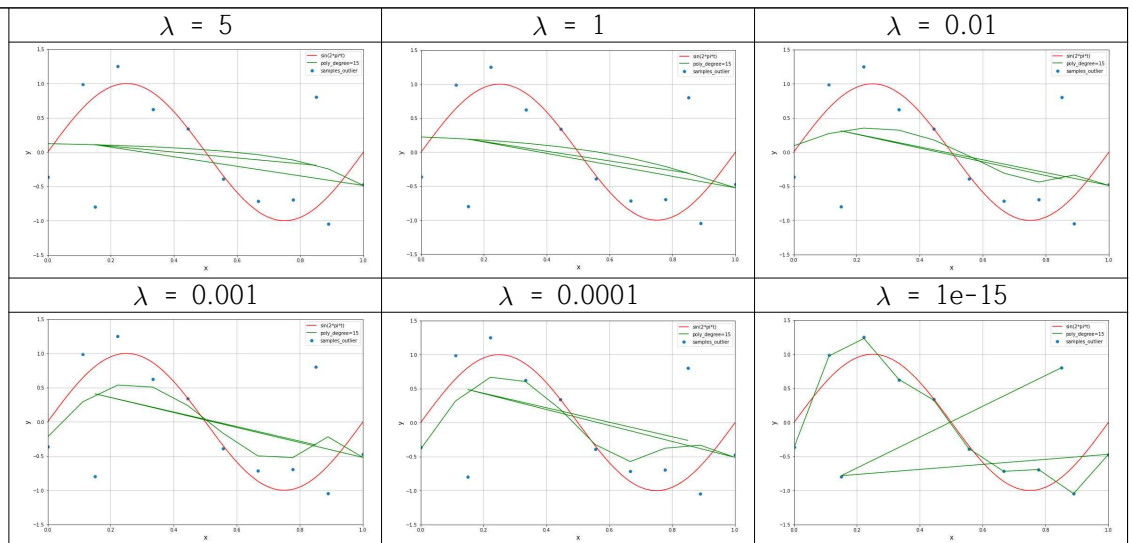
4번

polynomial
degree = 9



- polynomial degree = 9에서 L2규제를 적용하여, λ 값을 다양하게 준 결과

polynomial
degree = 15



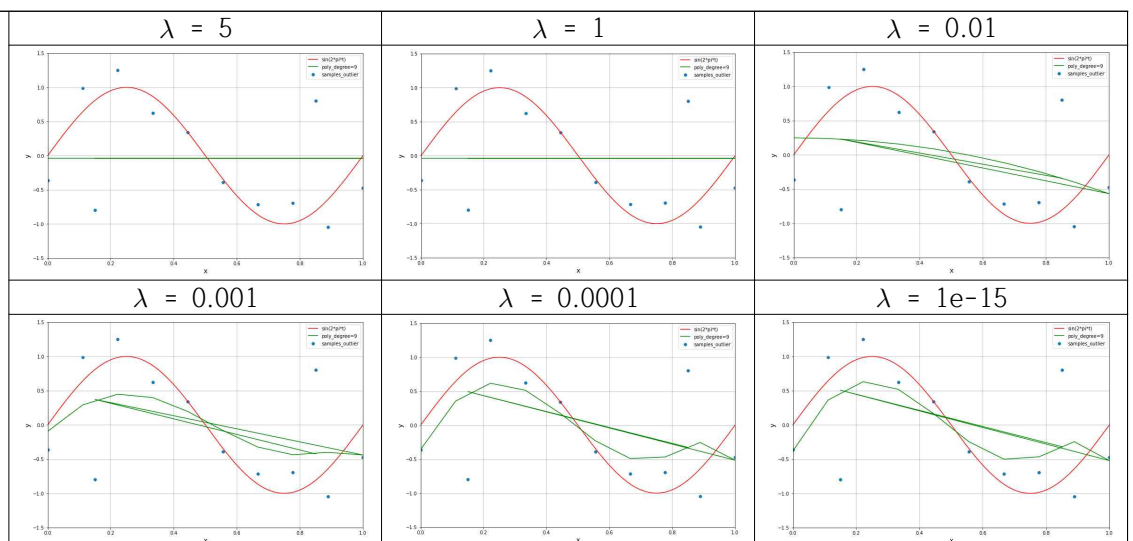
- polynomial degree = 15에서 L2규제를 적용하여, λ 값을 다양하게 준 결과

○ 주요 코드 <L2(Ridge) 규제>

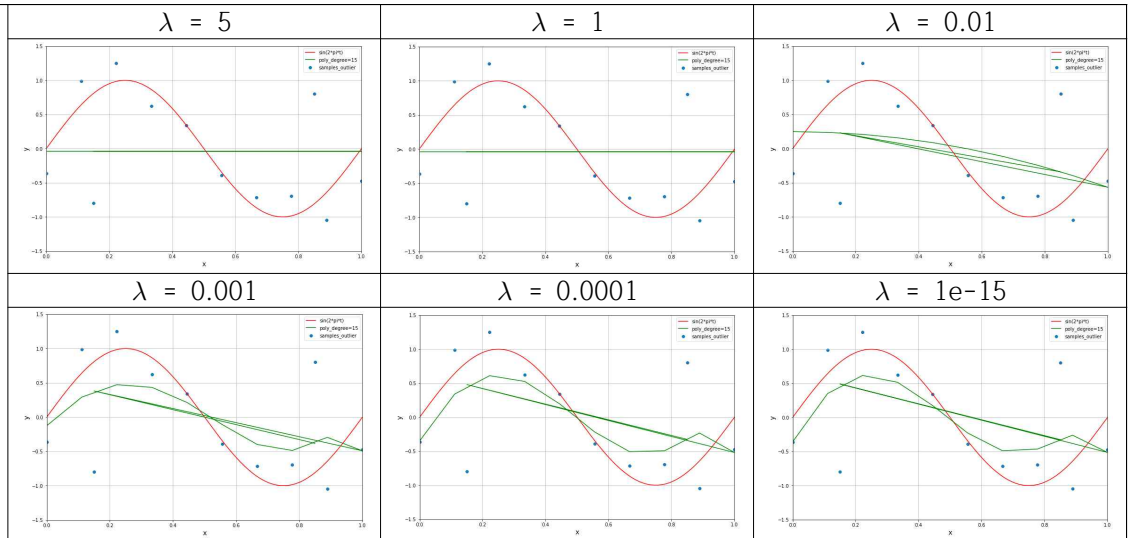
```
1 #L1 규제(lasso)
2 from sklearn.linear_model import Lasso
3 rcParams['figure.figsize'] = 12,8
4
5 poly_features = PolynomialFeatures(degree=9, include_bias=False)
6 x_poly = poly_features.fit_transform(x_s_e.reshape(-1,1))
7
8
9 lasso = Lasso(alpha=5).fit(x_poly, y_s_e)
10
11 plt.plot(x, y, color = 'r', label = 'sin(2*pi*x)')
12 plt.plot(x_s_e, lasso.predict(x_poly), color = 'g', label = 'poly_degree=9')
13 plt.scatter(x_s_e, y_s_e, label = 'samples_outlier')
14 plt.legend()
15
16 plt.xlim(0,1)
17 plt.ylim(-1.5,1.5)
18 plt.xlabel("x", size=14)
19 plt.ylabel("y", size=14)
20 plt.grid()
21
22 plt.show()
```

○ 분석 결과 <L1(Lasso) 규제>

polynomial
degree = 9



polynomial
degree =15



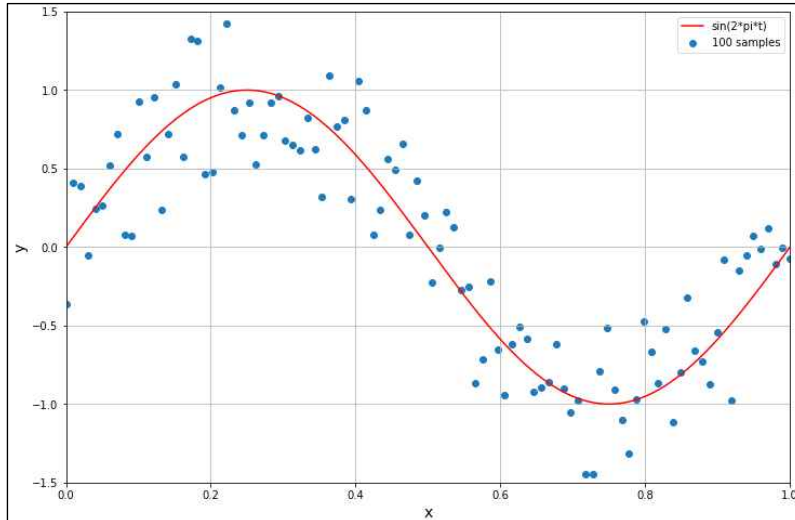
○ 분석 내용

- L2 규제의 λ 값이 커질수록 polynomial degree = 9에도 불구하고, 1차 linear regression과 비슷한 결과를 냈음을 확인할 수 있다.
- 반대로, L2 규제의 λ 값이 작을수록 규제가 적어 기존의 polynomial degree = 9의 regression line과 비슷한 그래프를 그려낸 것을 확인할 수 있었다.
- L2 규제와 L1 규제를 비교했을 때, 같은 차수와 같은 λ 값이 주어졌을 때 L1 규제가 과대적합이 되지 않도록 모델을 더 제한하는 모습을 보였다.
- L2 규제에서 λ 값을 매우 작게 했을 때, 기존의 regression line과 비슷해졌다. L1 규제는 영향력 없는 계수를 0으로 만들기 때문에 λ 값을 매우 작게 해도 기존의 regression line과 비슷해지지 않음을 확인할 수 있었다.

○ 주요 코드

```
11 def sin(x):      # sin(x)
12 | return np.sin(x)
13
14 def cos(x):      # cos(x)
15 | return np.cos(x)
16
17 x = np.linspace(0, 1, 1000)
18 y = sin(2*pi*x)
19 x_s = np.linspace(0, 1, 100)
20 #y_s = sin(2*pi*x_s)+random.uniform(-0.5, 0.5)
21 y_s = [sin(2*pi*(k))+random.uniform(-0.5, 0.5) for k in x_s]
22
```

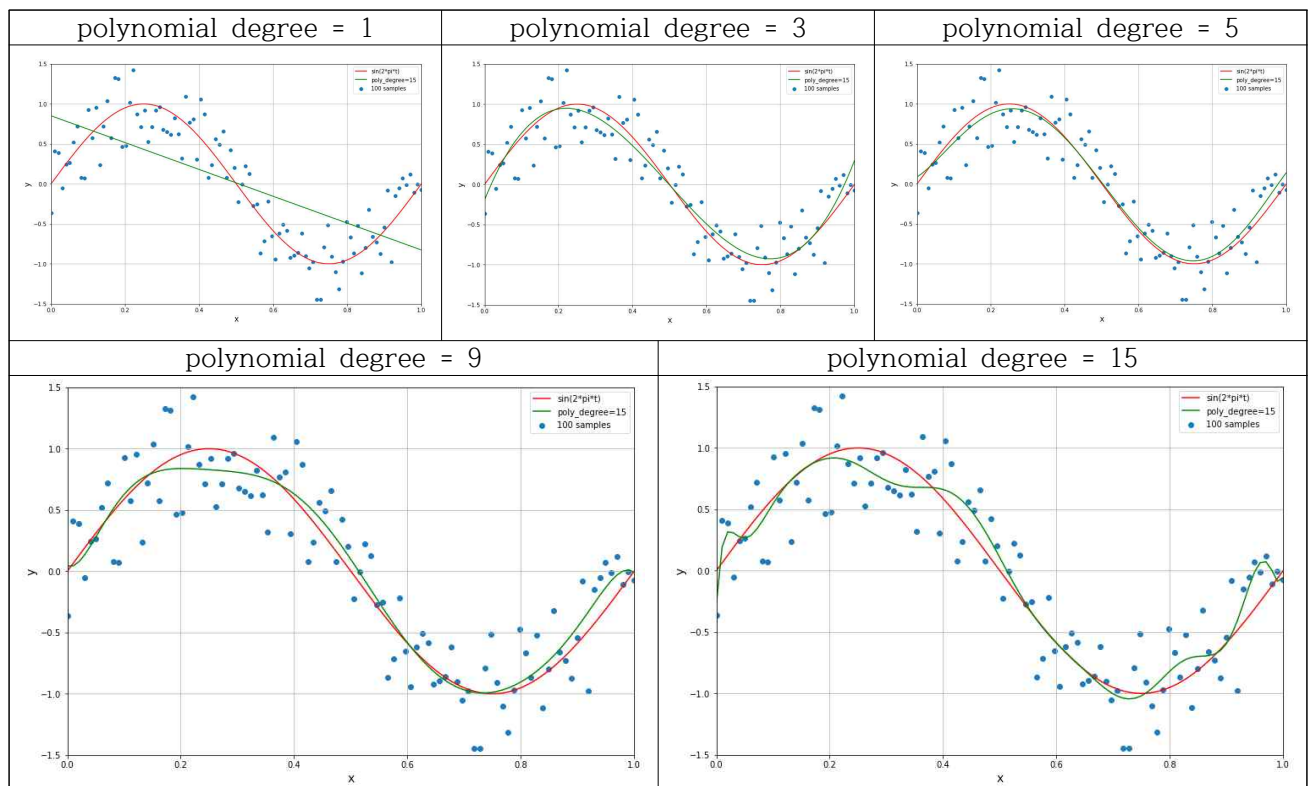

○ 분석 결과



- $\sin(2\pi x)$ 함수를 기반으로 Gaussian noise로 random.uniform 노이즈 -0.5~0.5를 부여해 plot한 결과
- 100개의 samples를 plot한 결과

○ 주요 코드

```
1 poly_features = PolynomialFeatures(degree=15, include_bias=False)
2 x_poly = poly_features.fit_transform(x_s.reshape(-1,1))
3
4
5 lr = LinearRegression().fit(x_poly, y_s)
6
7 plt.plot(x, y, color = 'r', label = 'sin(2*pi*t)')
8 plt.plot(x_s, lr.predict(x_poly), color = 'g', label = 'poly_degree=15')
9 plt.scatter(x_s, y_s, label = '100 samples')
10 plt.legend()
```



- | | |
|--|--|
| | <ul style="list-style-type: none">- 100 samples를 train하여 1, 3, 5, 9, and 15 degree에 따라 형성된 polynomial basis function을 만들고, regression lines를 그린 결과이다.- polynomial degree = 5와 9에서 overfitting되지 않고 sin함수와 가장 비슷한 형태의 그래프를 잘 학습하여 생성해낸 것을 확인할 수 있었다. |
|--|--|