CS302 Introduction to machine learning

School of Undergraduate Studies, DGIST

201611161 정원균

Homework report 1.

2022. 03. 27

Question 1. Plot 10 samples, spaced uniformly in range [0, 1], with the function sin(2πx) with a Gaussian noise

```
'''

1. Plot 10 samples, spaced uniformly in range [0, 1], with the function sin(2πx) with a
Gaussian noise

'''

import math
import random
import numpy as np
import matplotlib.pyplot as plt

pi = math.pi

x_list = np.linspace(0,1,10)
y_list = []
noise_list = np.random.normal(0,0.2,10)

y_list = np.sin(2*pi*x_list)+noise_list


X = np.linspace(0,1,200)
Y = np.sin(2*pi*X)

plt.plot(X,Y)
plt.scatter(x_list, y_list)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.xlim([0, 1])
plt.show()
```

그림 1. Code for Q1

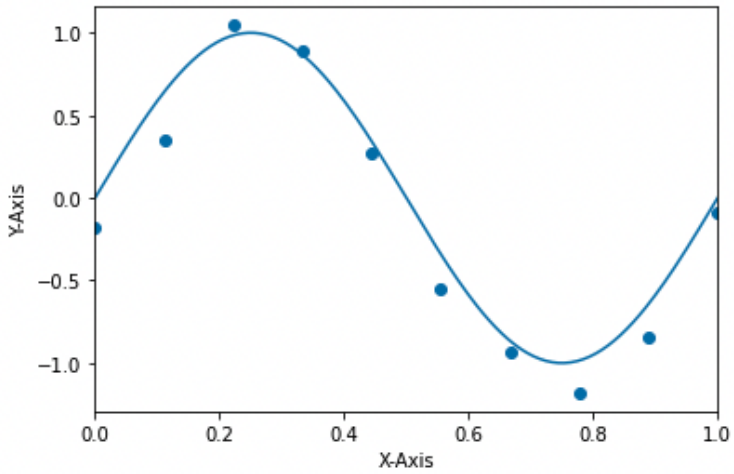

그림 2. Result for Q1

결과분석

위 결과와 같이 y = sin(2πx) model 에 np.random.normal function 을 이용해서 Gaussian noise 을 생성한 후에 sample 에 추가하였다.

Question 2. Generate regression lines with polynomial basis function with order 1, 3, 5, 9, and 15.

```
'''
 2. Generate regression lines with polynomial basis function with order 1, 3, 5, 9, and 15.
'''

from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures


LG = LinearRegression()
plt.figure(figsize = (15,10))

sample = [1,3,5,9,15]
n = 1

xp_list = x_list.reshape((-1,1))
yp_list = y_list

for i in sample:

    plt.subplot(2,3,n)

    poly = PolynomialFeatures(degree=i, include_bias=False)
    fitting_t = poly.fit_transform(xp_list)

    LG.fit(fitting_t,yp_list)
    poly_transform = poly.fit_transform(np.linspace(0,1,500).reshape(-1,1))


    plt.title("Order "+str(i)+" regression")
    plt.plot(np.linspace(0,1,500),LG.predict(poly_transform))
    plt.plot(X,Y)
    plt.scatter(x_list, y_list)
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.ylim([-3, 3])
    plt.xlim([0, 1])
    n+=1

plt.show()
```

그림 3. Code for Q2

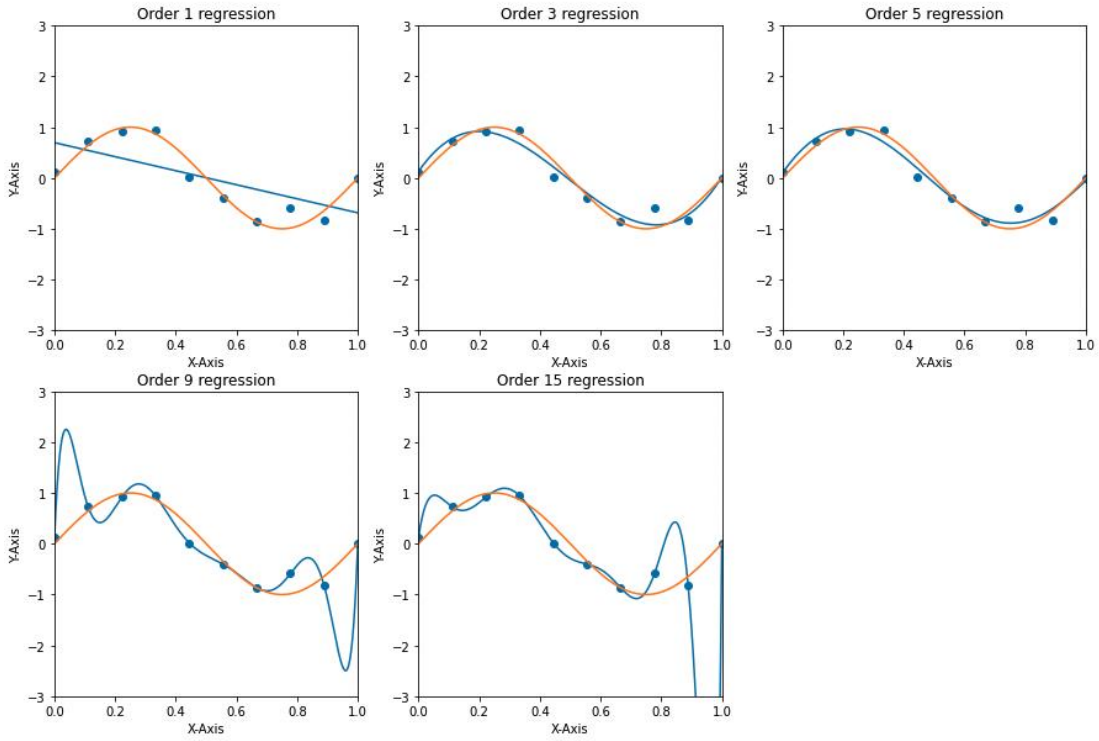

그림 4. Result for Q2

결과분석

Sklearn 의 LinearRegression 모듈과, Polynomial Feature 모듈을 이용해서 랜덤 샘플에 대한 각 order 별로 학습을 진행하여 Polynomial regression 의 결과를 나타내었다.

1. Order = 1 의 경우엔 제대로 샘플을 대표하는 Regression model 이 나오지 않는 Underfitting 현상이 나옴을 알 수 있다.

2. Order = 3, 5 의 경우 어느 정도 샘플을 대표할 수 있는 Regression model -> 샘플 생성 모델인 sin 함수와 비슷한 모델을 생성하였다.

3. Order = 9 부터 샘플에 과대 적합을 의미하는 Overfitting 현상이 발생함을 알 수 있고 Order = 15 일때 Order = 9 일때 보다 Overfitting 이 더 크게 발생하였다.

Question 3. Add 2 or 3 points of exceptional outliers that do not follow sin(2πx) and then generate regression lines with polynomial basis function with order 1, 3, 5, 9, and 15.

```
'''

3. Add 2 or 3 points of exceptional outliers that do not follow sin(2πx) and then generate regression lines with
polynomial basis function with order 1, 3, 5, 9, and 15.

'''
import numpy as np

x_outlier = [0.38,0.46,0.60]
y_outlier = [0.20,0.70,0.30]

new_x = np.append(x_list, np.array(x_outlier))
new_y = np.append(y_list, np.array(y_outlier))

plt.figure(figsize = (15,10))
sample = [1,3,5,9,15]
n = 1

xp_list = new_x.reshape((-1,1))
yp_list = new_y

for i in sample:

    plt.subplot(2,3,n)

    poly = PolynomialFeatures(degree=i, include_bias=False)
    fitting_t = poly.fit_transform(xp_list)

    LG.fit(fitting_t,yp_list)
    poly_transform = poly.fit_transform(np.linspace(0,1,500).reshape(-1,1))

    plt.title("Order "+str(i)+" regression")
    plt.plot(np.linspace(0,1,500),LG.predict(poly_transform))
    plt.plot(X,Y)
    plt.scatter(new_x, new_y)
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.ylim([-3, 3])
    plt.xlim([0, 1])
    n+=1

plt.show()
```

그림 5. Code for Q3

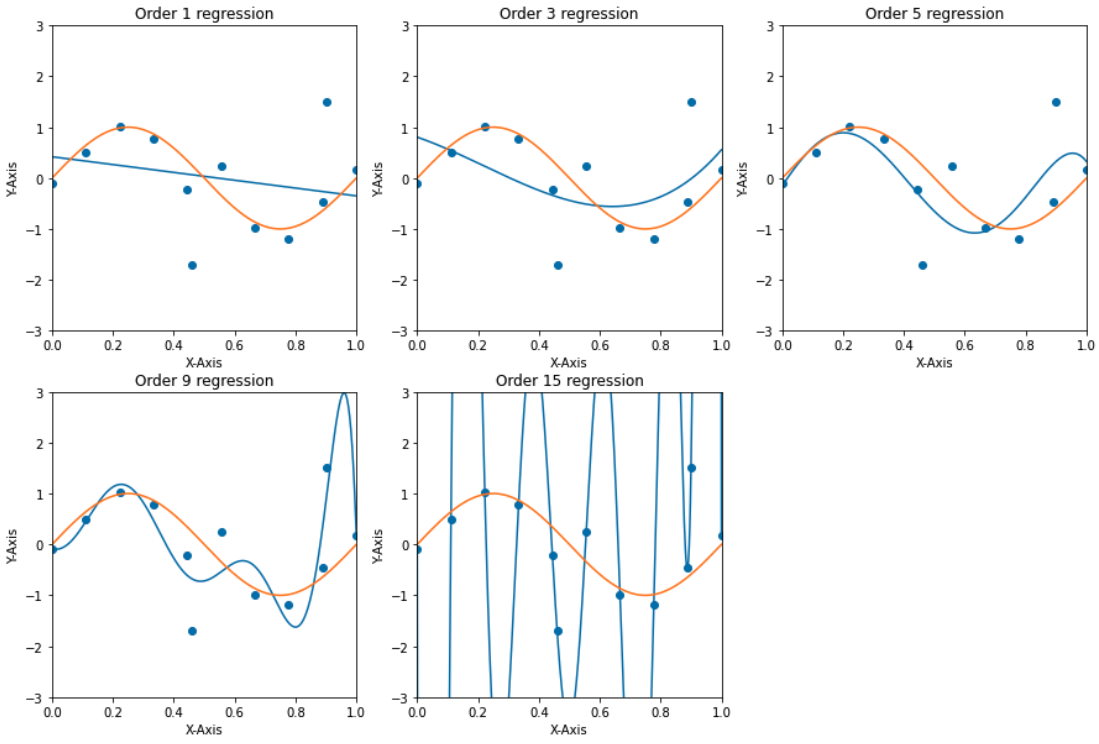

그림 6. Result for Q3

결과 분석

2 번 결과에 가설 모델을 따르지 않는 임의의 샘플을 추가한 다음 Sklearn 의 LinearRegression 모듈과, Polynomial Feature 모듈을 이용해서 랜덤 샘플에 대한 각 order 별로 학습을 진행하여 Polynomial regression 의 결과를 나타내었다.

1. Order = 1 의 경우엔 여전히 제대로 샘플을 대표하는 Regression model 이 나오지 않는 Underfitting 현상이 나옴을 알 수 있다.

2. Order = 3, 5 의 경우 어느정도 주어진 샘플을 따라가는 model 을 생성하는 것처럼 보이지만 일관성 있는 model 이 나오지 않고, 2 번 질문과 비교해보았을 때도 가설을 따르지 않는 임의의 샘플을 추가하였을 때 튀는 값들 때문에 제대로 fitting 이되지 않았음을 알 수 있다.

3. Order = 9 부터 샘플에 과대 적합을 의미하는 Overfitting 현상이 발생함을 알 수 있고 Order = 15 일때 Order = 9 일때 보다 Overfitting 이 더 크게 발생하였다. 2 번 질문과 비교해 보았을 때 훨씬 더 큰 과적합을 보임을 알 수 있고 특히 Order 가 15 일 경우 거의 제대로 fitting 을 못 하는 것을 볼 때 임의의 노이즈가 추가되었고, 차수가 높아질 수록 과적합현상이 심해져 제대로 샘플을 대표하는 모델이 나오지 않음을 알 수 있다.

Question 4. For the case including the outliers, generate the regression lines with the L2 regularization term with order 9 and 15. Show how the lines are changed with respect to λ. Generate the regression lines with the L1 regularization term and compare the lines with L2 regularization.

```
'''

4. For the case including the outliers, generate the regression lines with the L2 regularization
term with order 9 and 15. Show how the lines are changed with respect to λ. Generate the
regression lines with the L1 regularization term and compare the lines with L2 regularization.

'''

from sklearn.linear_model import Lasso,Ridge

plt.figure(figsize = (15,10))
plt.suptitle('L1 norm restriction with order 9',fontsize=20)

n = 1
lambda_list = [1.0,0.5,0.10,0.05,0.010,0.005,0.001,0.0005,0.0001]

for i in lambda_list:


    plt.subplot(3,3,n)
    plt.subplots_adjust(left=0.125, bottom=0.1,  right=0.9, top=0.9, wspace=0.2, hspace=0.35)

    xn_list = x_list.reshape((-1,1))
    yn_list = y_list

    poly = PolynomialFeatures(degree=9, include_bias=False)
    fitting_t = poly.fit_transform(xn_list)

    lasso = Lasso(alpha = i, max_iter = 100000)
    lasso.fit(fitting_t,yn_list)

    poly_transform = poly.fit_transform(np.linspace(0,1,500).reshape(-1,1))

    p = str(i)
    plt.title("lambda = "+ p)
    plt.plot(np.linspace(0,1,500),lasso.predict(poly_transform))
    plt.plot(X,Y)
    plt.scatter(x_list, y_list)
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.ylim([-3, 3])
    plt.xlim([0, 1])
    n+=1

plt.show()
```

그림 7. Code for Q4, L1 norm

```
plt.figure(figsize = (15,10))
plt.suptitle('L2 norm restriction with order 9',fontsize=20)

n = 1

for i in lambda_list:

    plt.subplot(3,3,n)
    plt.subplots_adjust(left=0.125, bottom=0.1,  right=0.9, top=0.9, wspace=0.2, hspace=0.35)

    xn_list = x_list.reshape((-1,1))
    yn_list = y_list

    poly = PolynomialFeatures(degree=9, include_bias=False)
    fitting_t = poly.fit_transform(xn_list)

    ridge = Ridge(alpha = i, max_iter = 100000)
    ridge.fit(fitting_t,yn_list)

    poly_transform = poly.fit_transform(np.linspace(0,1,500).reshape(-1,1))

    p = str(i)
    plt.title("lambda = "+ p)
    plt.plot(np.linspace(0,1,500),ridge.predict(poly_transform))
    plt.plot(X,Y)
    plt.scatter(x_list, y_list)
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.ylim([-3, 3])
    plt.xlim([0, 1])
    n+=1

plt.show()
```

그림 8. Code for Q4, L2 norm

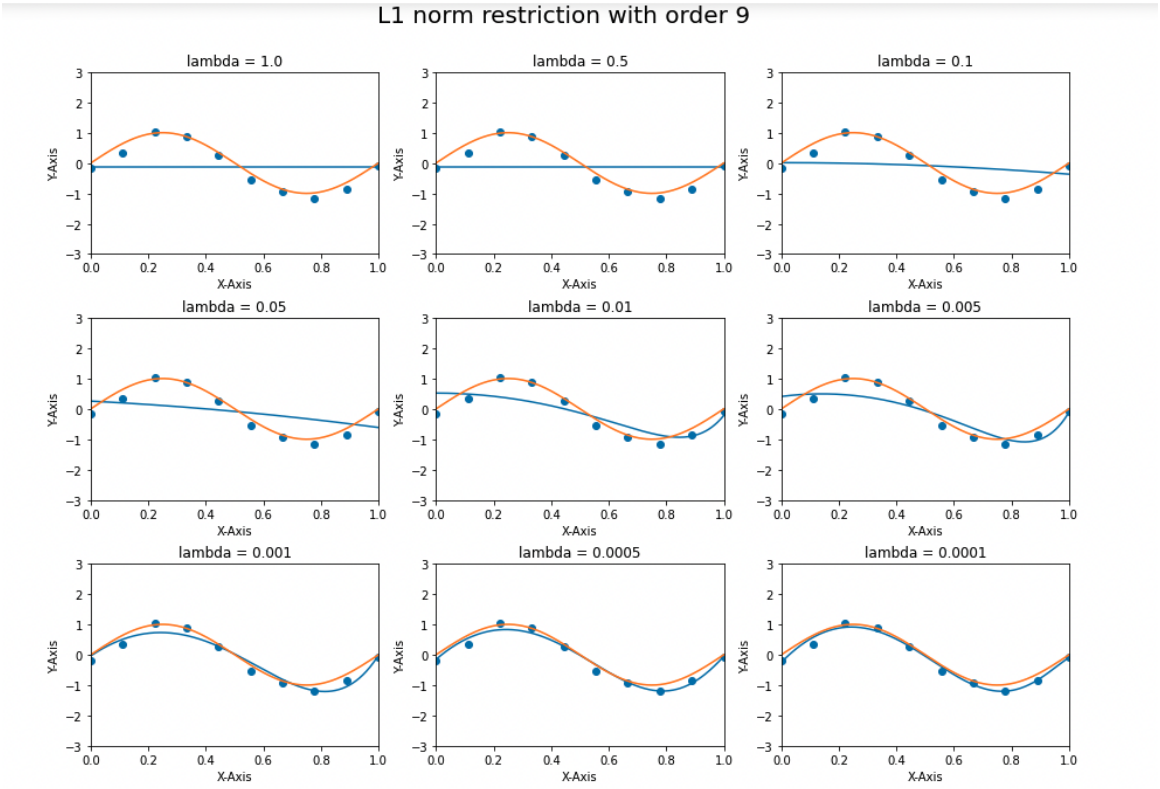

그림 9. Result for Q4, L1 norm

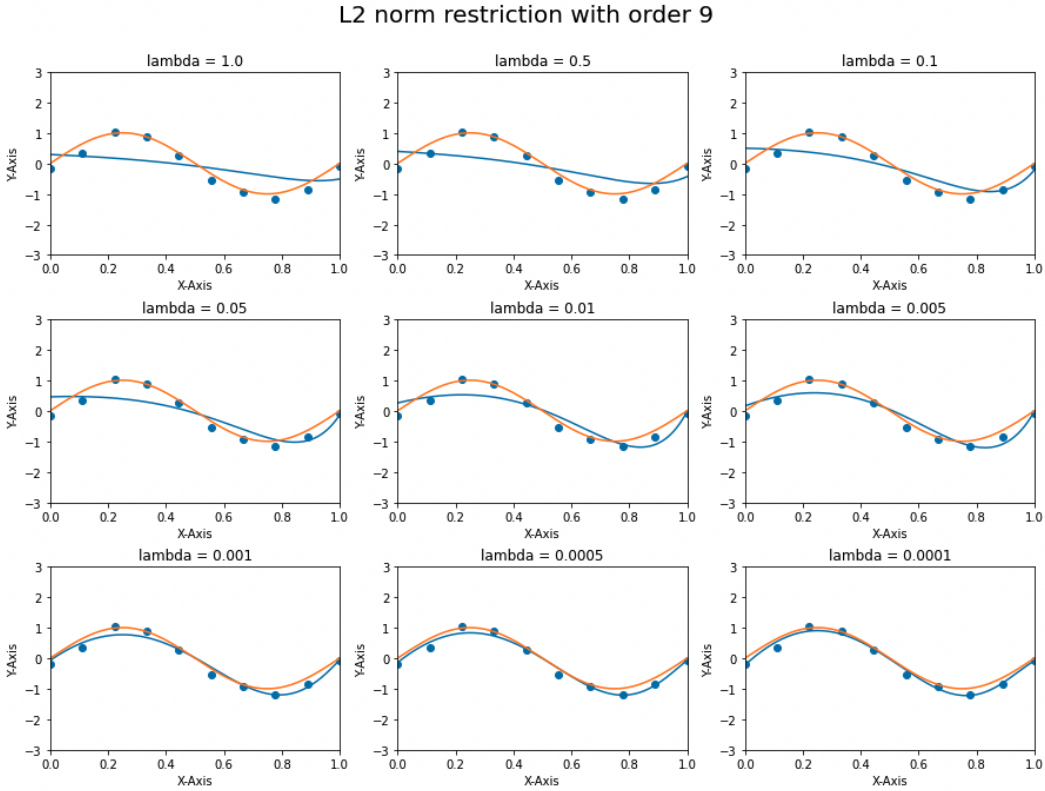

그림 10. Result for Q4, L2 norm

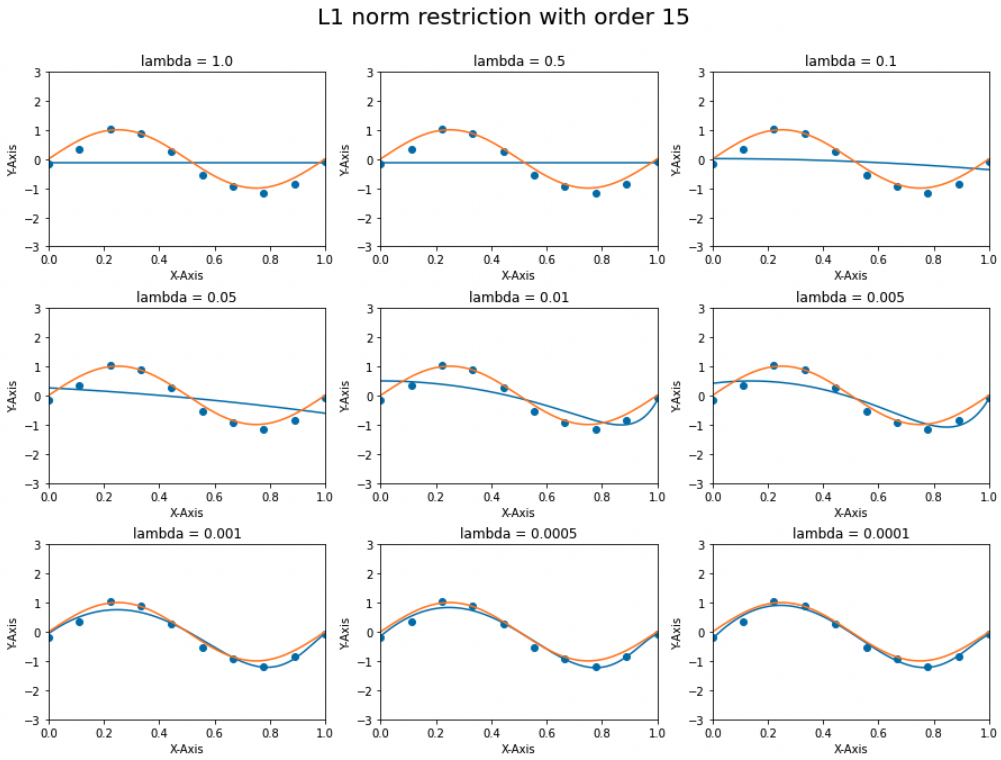

그림 11. Result for Q4, L1 norm

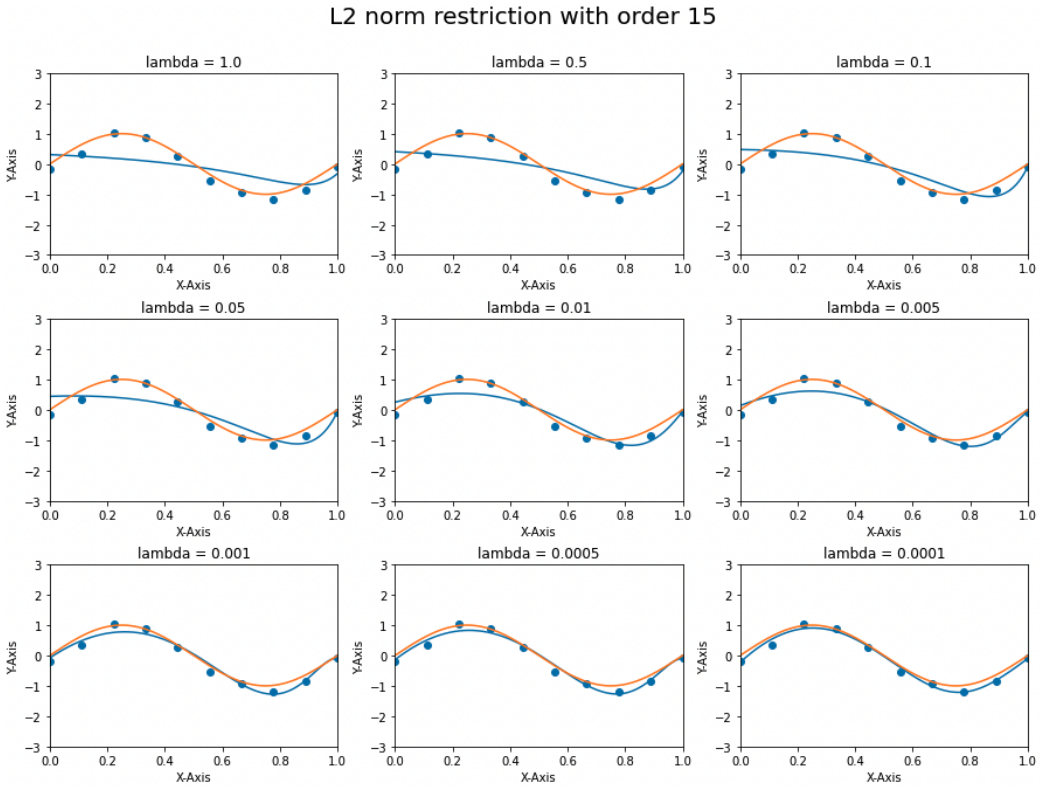

그림 12. Result for Q4, L2 norm

결과 분석

1. L_1 Regularization

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

일반적인 Linear Regression Cost Function 에서 weighted value 을 두는데 L1 norm 의 경우엔 절대값 가중치를 주어 Cost Function 을 편미분하는 과정에서 가중치가 작은 값은 상수 값에 의해서 weighted 가 0 이 되어서 몇몇 중요한 가중치만 남기게 된다.

2. L_2 Regularization

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

Loss function

Regularization Term

L2 norm 의 경우엔 Cost function 에 제곱한 가중치 값을 더해줌으로써 편미분을 통해 Cost 뿐만 아니라 가중치 또한 줄어드는 방식으로 학습을 한다. 특정 가중치가 비이상적으로 커지는 상황을 방지하여 전체적으로 가중치를 작아지게 하고 과적합을 방지한다.

L1, L2 cost function 에서 λ가 Regulation 의 가중치를 주는 역할을 한다. 따라서 λ값이 작아질 수록 Regulation Term 이 증가하여 모델이 복잡해지고(overfitting 의 위험) λ값이 커질 수록 단순한 모델(underfitting 의 위험)이 나타난다.

특히 L1 의 경우 중요하지 않은 feature 의 가중치를 0 으로 만들어 버리기 때문에 λ가 1 일때 거의 모델이 학습되지 않음을 알 수 있고 L2 의 경우엔 0 으로 만들진 않고 0 과 가까운 값으로 바꾸기 때문에 L1 만큼 극단적인 과소 적합 모델이 나오지 않는다.

또한 L1 는 λ가 0.001 부터 적절한 fitting 이 일어나고 L2 는 λ가 0.01 부터 적절한 fitting 이 이루어지는 것을 보았을 때 이 모델의 경우엔 L2 Regulation 을 통한 모델이 조금 더 fitting 을 빨리 함을 알 수 있다.

Question 5. Plot 100 samples with the function sin(2πx) instead of 10 samples, and then generate the regression lines with order 1, 3, 5, 9, and 15.

```
'''

5. Plot 100 samples with the function sin(2πx) instead of 10 samples, and then generate the regression lines with
order 1, 3, 5, 9, and 15.

'''

x_100 = np.linspace(0,1,100)
noise_list = np.random.normal(0,0.2,100)
y_100 = np.sin(2*pi*x_100)+noise_list

'''plt.plot(X,Y)
plt.scatter(x_100, y_100)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.ylim([-1.5, 1.5])
plt.xlim([0, 1])
plt.show()'''


plt.figure(figsize = (15,10))
sample = [1,3,5,9,15]
n = 1

for i in sample:
    xp_list = x_100.reshape((-1,1))
    yp_list = y_100
    plt.subplot(2,3,n)

    poly = PolynomialFeatures(degree=i, include_bias=False)
    fitting_t = poly.fit_transform(xp_list)

    LG.fit(fitting_t,yp_list)
    ploy_transform = poly.fit_transform(np.linspace(0,1,1000).reshape(-1,1))


    plt.title("Order "+str(i)+" regression")
    plt.plot(np.linspace(0,1,1000),LG.predict(ploy_transform),color='red')
    plt.plot(X,Y)
    plt.scatter(x_100, y_100)
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.ylim([-1.5, 1.5])
    plt.xlim([0, 1])
    n+=1
plt.show()
```

그림 13. Code for Q5

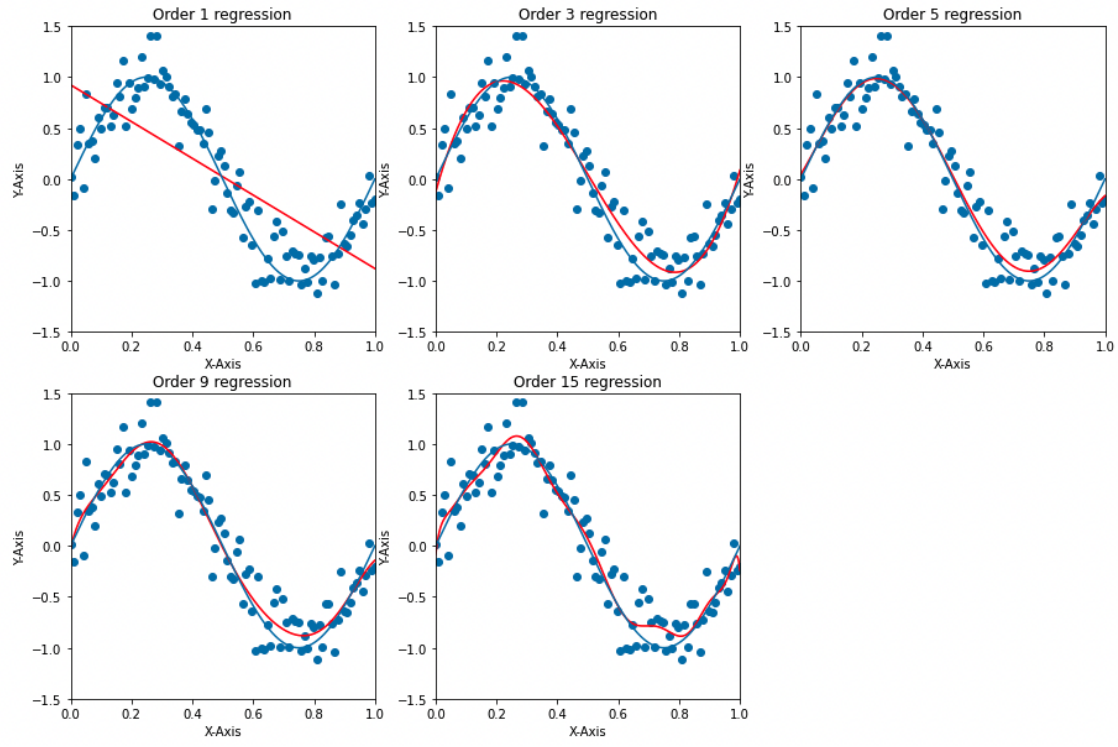

그림 14. Result for Q5, code

결과 분석

1 번 문제와 같은 방법으로 $y = \sin(2\pi x)$ 모델에 Gaussian Noise 을 추가한 100 개의 샘플을 나타내었다. 파란 그래프의 경우 $y = \sin(2\pi x)$을 붉은 그래프의 경우에 Polynomial Regression 을 적용한 결과이다.

1. Order = 1 일 때 샘플의 개수가 10 개인지 100 개인지에 관계없이 샘플 들을 제대로 표현하지 못하는 Underfitting 이 발생함을 알 수 있다.

2. Order = 3, 5 일 때 1 번 문제에 대한 결과와 비슷하게 $y = \sin(2\pi x)$ 모델과 비슷한 학습결과를 보여주고, 1 번에 비해서 조금 더 생성 모델과 비슷한 Regression Line 이 나옴을 알 수 있다.

3. Order = 9, 15 일 때 1 번 문제와 가장 큰 차이점이 발견된다. 1 번 결과에서 Order = 9 이상일 때부터 Overfitting 이 발생하였는데 샘플의 개수가 100 개로 늘었을 때 Overfitting 현상이 감소함을 알 수 있고 Order = 15 일 때 아직 조금의 Overfitting 이 발생하지만 샘플의 개수가 더 늘어날 경우 줄어들 가능성이 많다고 볼 수 있다.

-End of Report -