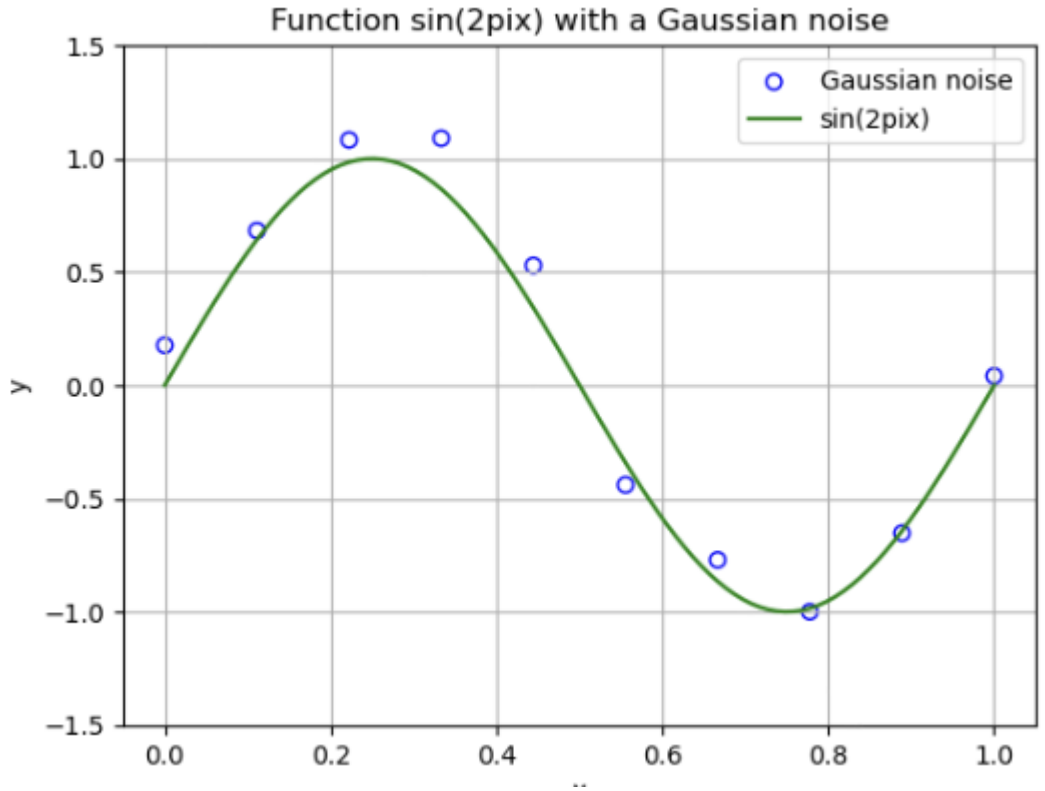
기계학습개론 과제1 리포트

202011047 김승태

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

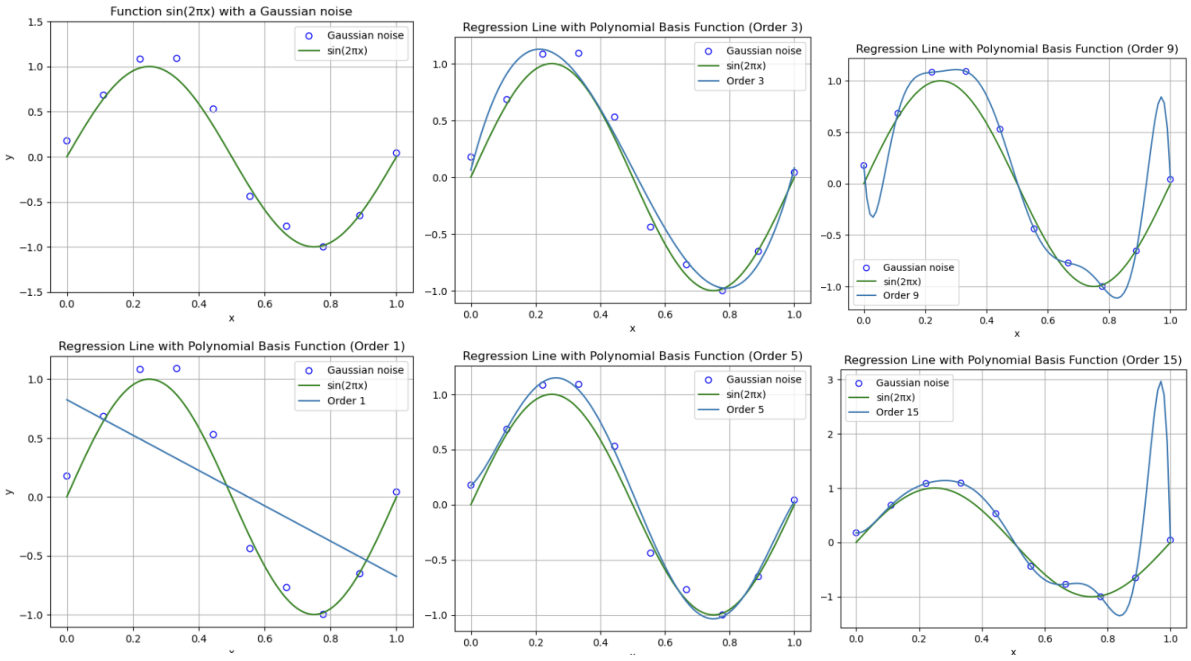
- 코드 설명 :일단, numpy 패키지를 import하고, pyplot 모듈을 plt로 가져오고, random seed를 생성한다. np.random.seed(0)로 난수 생성해서 매번 달라지지 않도록 고정한다. 0,1,2,3, .. 를 해 보았지만, 0의 결과가 가장 과제 예시 그림에서 보여주려 한 것과 결과값이 비슷하게 나올 것 같아 0을 채택하였다. Linspace로 등간격 10개 점을 찍어 표현하고, gaussian_noise에 random.normal을 사용하여, 저장하여서 y값에 그 값을 더하는 것으로 noise를 주게 설계하였다. 결과 그림은 다음과 같다.



결과 분석 : 일단, 1번은 예시 그림에 최대한 비슷하게 그리려 노력하였으므로, 위와 같이 되었다. 별건 아니지만, 색깔과 noise 모양도 비슷하게끔 코드를 제작하였다.

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

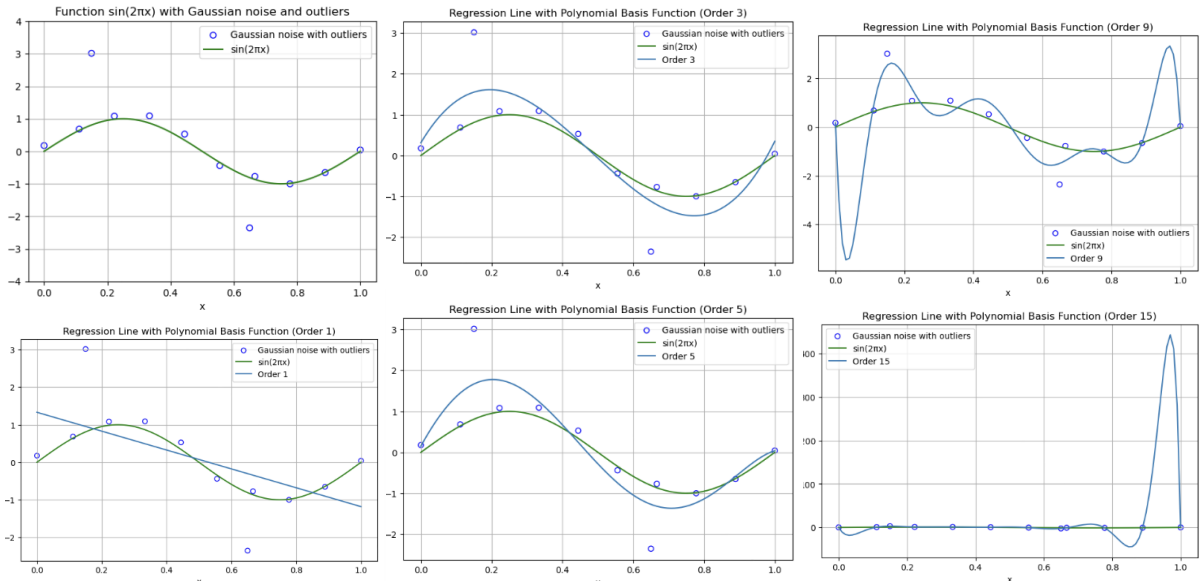
- 코드 설명 : gen_regression_line을 추가하여, Regression line with polynomial basis function을 제작하고자 한다. PolynomialFeatures로 다항식 특성을 생성하고, 차수에 따라 regression 모델을 학습하게 된다. 가장 중요한 것은, sklearn.linear_model에서 LinearRegression을 import하여 사용한다는 것이다. 그리고 y_pred = lin_reg.predict(poly_features.transform(np.linspace(0, 1, 100).reshape(-1, 1))) 이 줄에서, 예측 y를 계산해서 plot시키면 된다. 결과는 다음과 같다.



결과 분석 : order가 1일 때에는, 예측을 제대로 하지 못하는 underfitting 현상이 일어나게 된다. 그러다가, Order 3에서는 어느 정도 예측을 잘 하는 것을 볼 수 있다. Order 5에서는 가장 원 함수인 2πx와 비슷한 모양새를 띄게 된다. 그렇지만, order 9 부터는, 모든 점을 지나가는 overfitting이 일어나면서, 오히려 원 함수와는 먼 쪽의 모양새를 가지게 된다. 15에서는 오히려 중간 지점이 안정적이게 되지만, 마지막에는 갑자기 값이 튀는 것으로 보아, overfitting이 확실함을 알 수 있다.

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.

-코드 설명 : 2번과 같은 코드에서, x_outliers, y_outliers를 추가하고, 그 것을 concatenate로 추가한다. 각각의 y값을 3.0 , -2.5로 해서 기존 값과 확실한 차이를 두지만, 전체적 개형이 변하지는 않도록 조정하였다.(order가 낮을 때) 결과값은 다음과 같다.

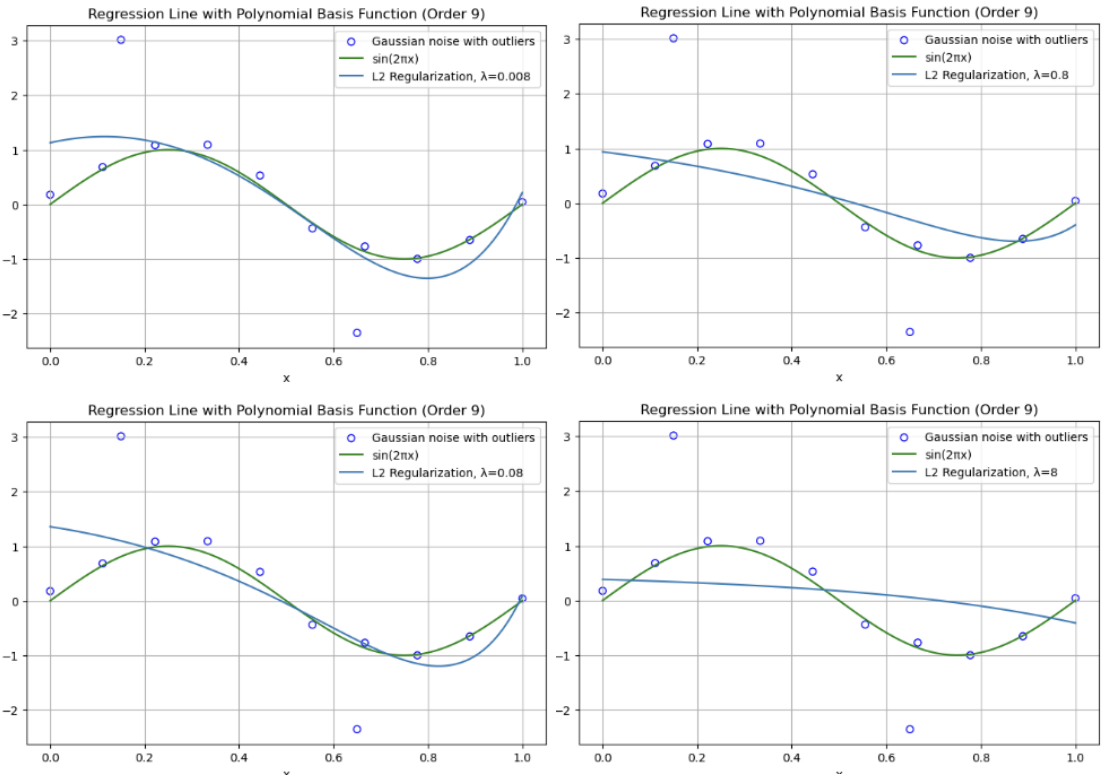


-결과 분석 : outlier들을 추가했을 때가 추가하지 않았을 때에 비해서, 확실히 눈에 띄게 극적인 예측을 하는 것을 볼 수 있었다. 이번에도 역시나, 3, 5 정도가 가장 이상적으로 보이고, 1 은 underfitting, 9, 15는 overfitting됨을 볼 수 있었다. 특히, order 15는 굉장히 이상한 값을 띄었다. 거의 y 값이 몇 백을 넘어버리는 사태까지 이르게 되었다.

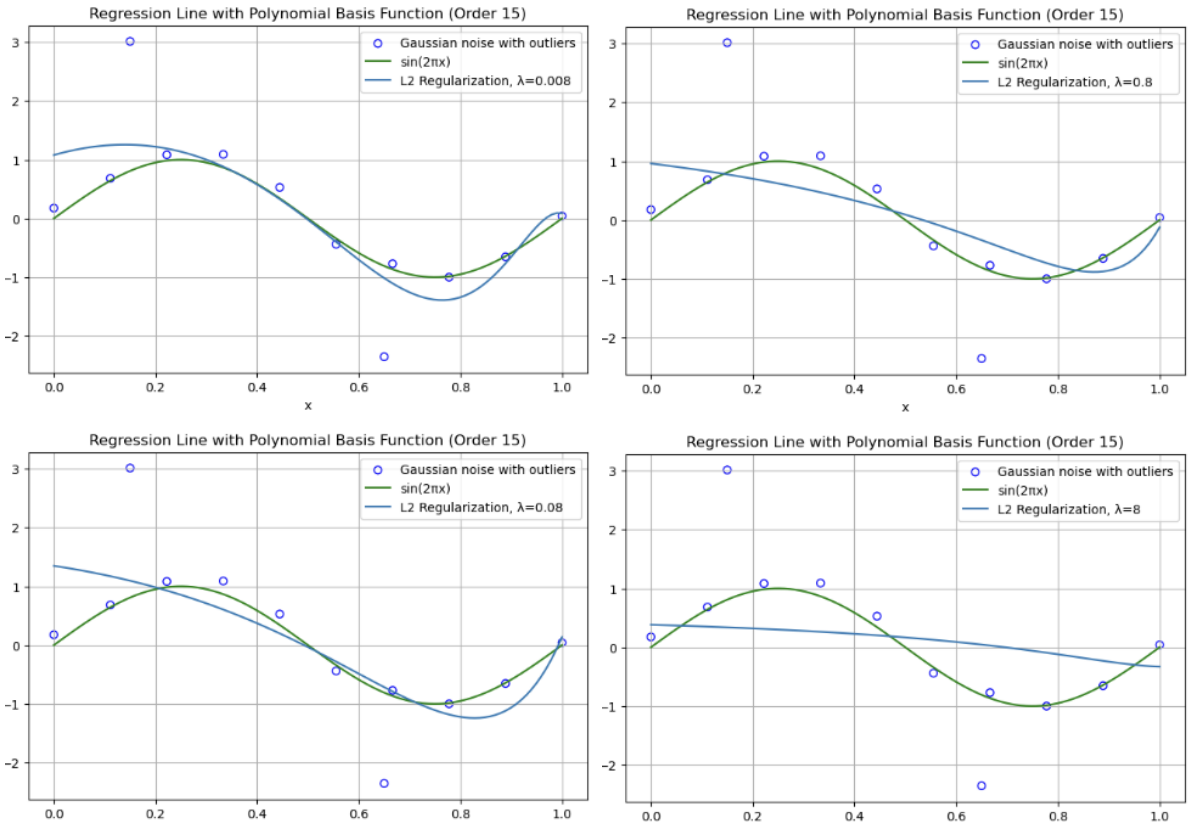
4.For the case including the outliers, generate the regression lines with the L2 regularization term with order 9 and 15. Show how the lines are changed with respect to lambda. Generate the regression lines with the L1 regularization term and compare the lines with L2 regularization.

-코드 설명 : 이번에는 2번과 달리, LinearRegression을 쓰지 않고, Ridge, Lasso를 사용하여 L1 regularization과 L2 regularization을 사용해 보도록 하였다. 이전 문제들의 코드를 붙여넣기 하여 사용하였고, LinearRegression은 추후에 사용할 수 있다는 생각을 가지고, 아예 사용하지 않지는 않고, else문으로 따로 빼 두었다. outlier들을 그대로 두고 진행하였다. Gen_regression_line 함수에 regularization이라는 인자를 두고 각각 L2, L1 에 따라 L2는 Ridge, L1은 Lasso를 사용하여 model를 바꾸어서 regression line을 생성하였다. 또한, order를 9, 15로 지정하여서 결과값을 지켜보았다. 또한, lambdas 를 지정하였는데, 실행 결과, 0.001부터 실행하였을 경우, time 지연이 너무 많이 되는 탓이었는지 오류가 발생하여서, 조정한 결과, 0.008부터 유효한 결과값을 보여주었기 때문에, 0.008 값부터, 10배씩 하여 8까지 증가시키며 진행하였다. 결과값은 다음과 같다.

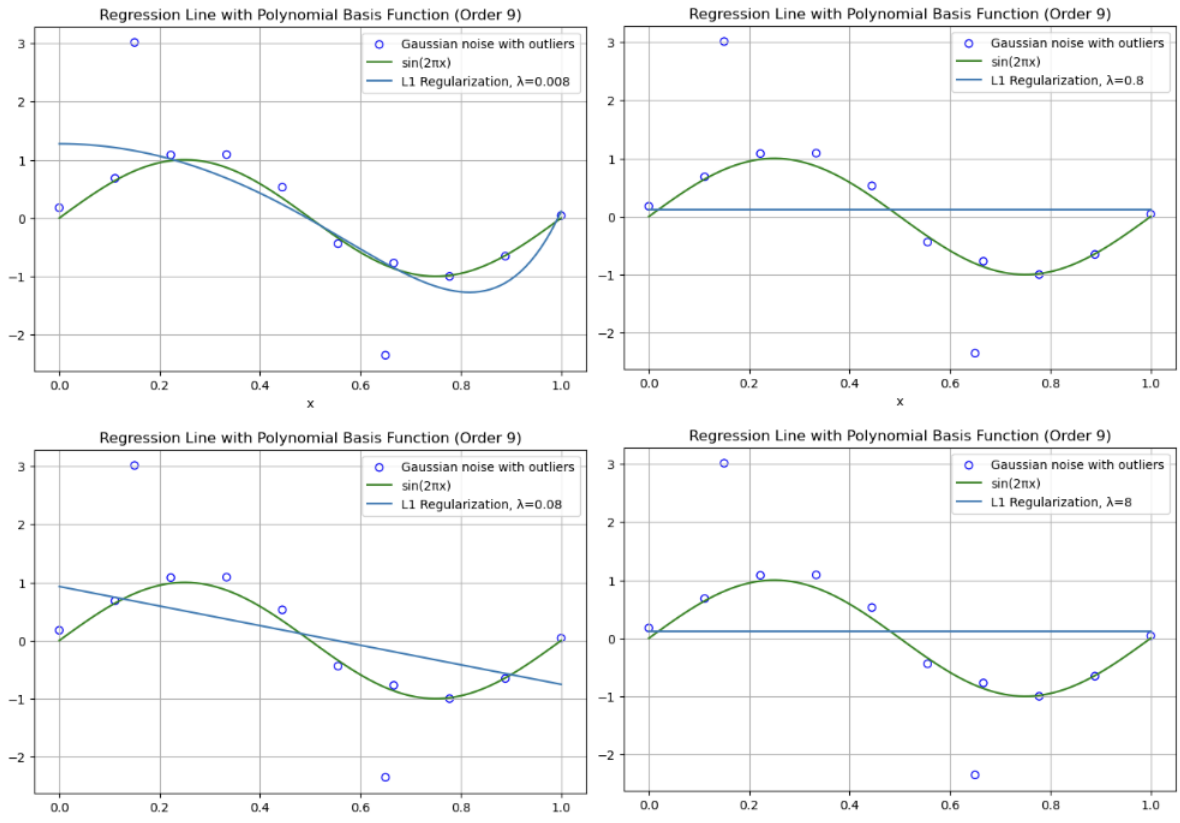
L2, order 9



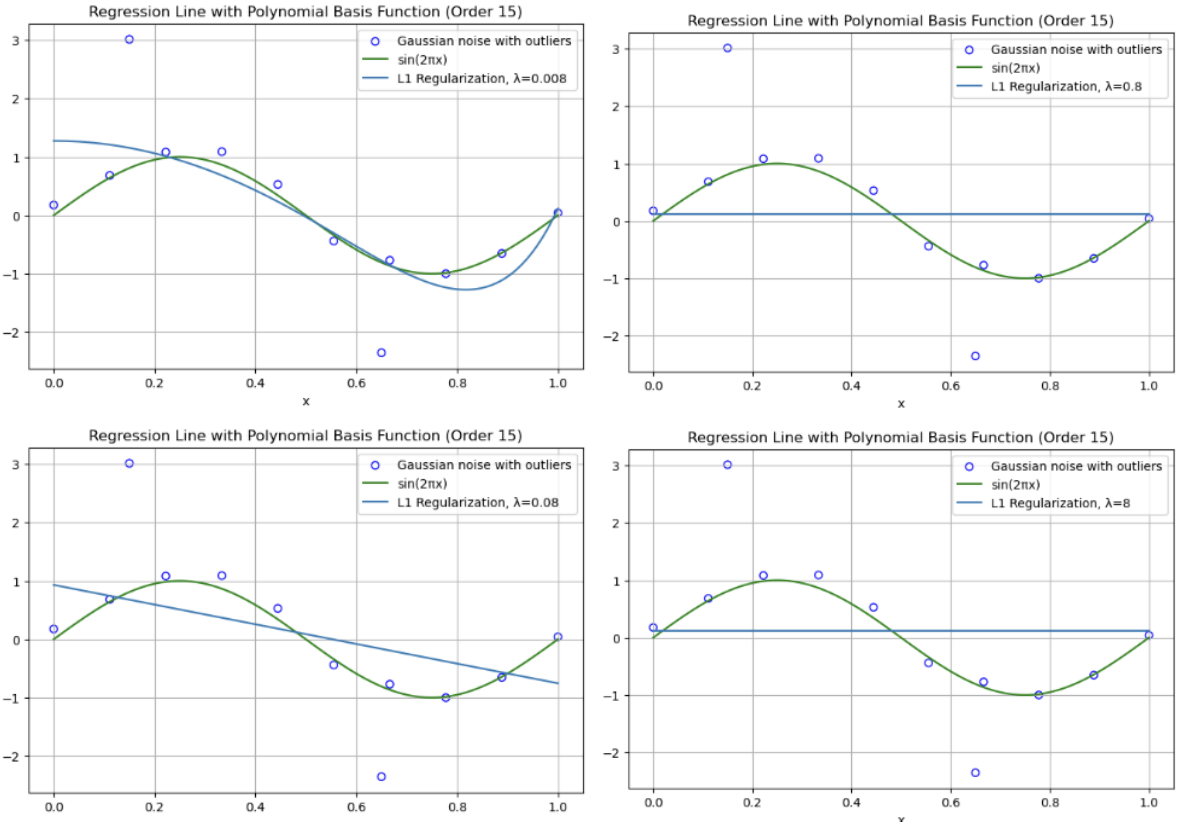
L2, order 15



L1, order 9



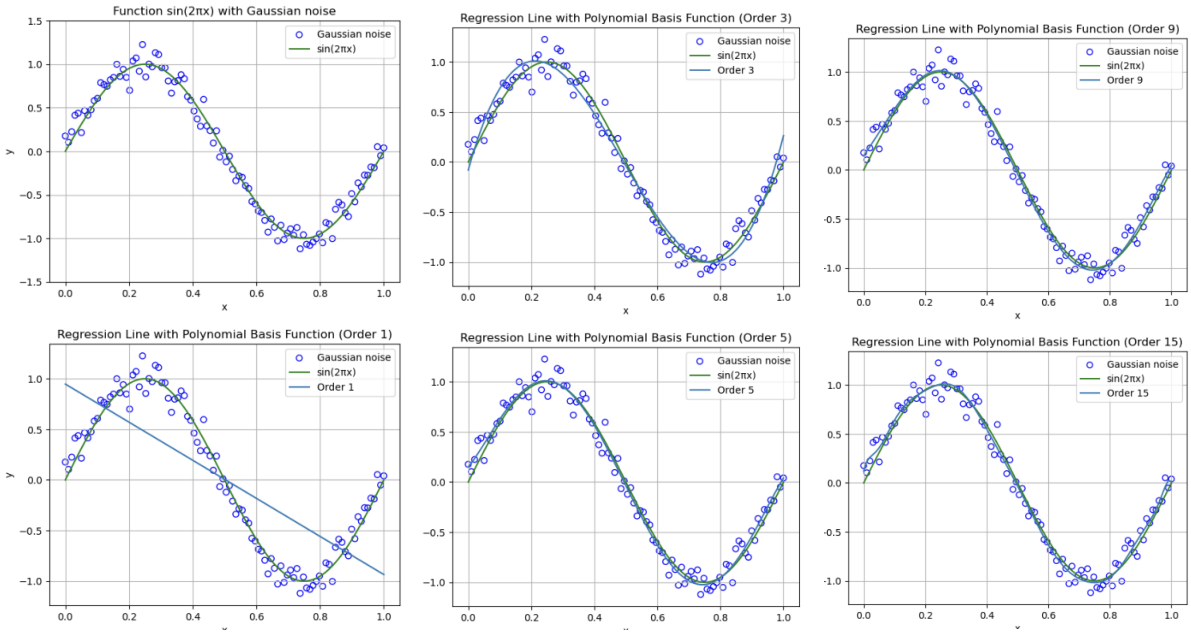
L1, order 15



-결과 분석 : 코드 결과를 보면서 가장 놀라웠던 점은, order 9, 15에 대해서 거의 차이가 없다는 것이었다. 그만큼 Regularization이 잘 되었다고 볼 수 있는데, 정규화시키면서 overfitting을 확실히 줄일 수 있다는 것을 보여 주었다. 즉, L2, L1 모두 이러한 점에서는 효과적으로 제어하고 있다는 것이 확실해 진 것이다. 하지만, L2, L1에는 차이점이 있다. 둘 다 가중치에 기반하여 페널티를 부여하는 방식인데, L1은 절댓값으로, L2는 제곱에 비례하여 페널티를 부여하여, L1이 좀 더 과격하게 가중치 크기를 줄여 0으로 만들 것이고, L2는 0에 가깝게만 만들 것이어서 조금 더 과격하지 않게 결과가 나온 것으로 보인다. 실제 결과값에서 L1은 거의 가중치가 0가 되어서, 람다 값이 1이 넘기도 전에 0.8부터 벌써 x축 평행 일직선 형태의 regression line이 형성되게 된다. 그나마 L2는 어느 정도까지는 남아 있는데 비해, L1이 더 과격하게 가중치를 줄인다는 것을 증명한 것이다.

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.

-코드 해석 : 이 것은 2번에서 sample 수만 늘린 것이므로, 2번 코드에서 sample 수만 바꾸면 된다. 즉, x=np.linspace(0,1,10)을 x=np.linspace(0,1,100)으로 바꾸기만 하면 된다는 것이다. 결과값을 보겠다.



-결과 분석 : 결과적으로 보면, order 1을 제외하고는 거의 비슷한 양상을 띄게 되는 것을 살펴볼 수 있다. Order 1은 일차 그래프여서 양상을 애초에 담을 수 없는 것을 제외하고, 그 이상의 order 3부터 살펴보겠다. 지금 상황에서는 order보다 sample의 개수가 많기 때문에, overfitting의 확률이 적은 것을 확실히 볼 수 있었다. 튀는 그래프도 없었고, 대략적으로 흐름을 잘 따라가는 양상을 보였다. 이를 통해, sample의 개수가 늘어나면, regression을 더 잘 할 수 있다는 것을 효과적으로 확인할 수 있었다.