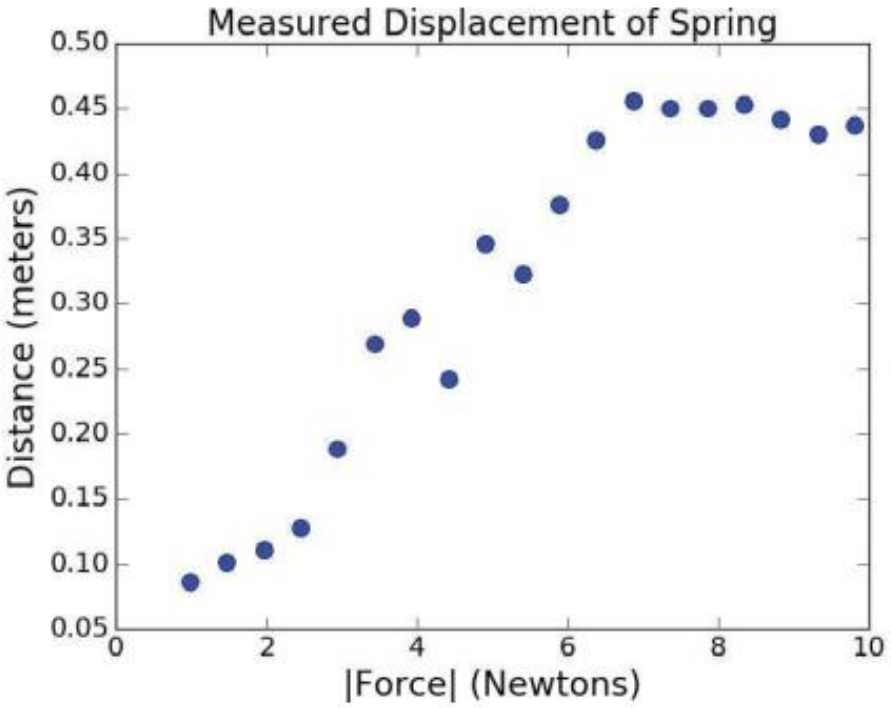
Understanding Experimental Data

김이언

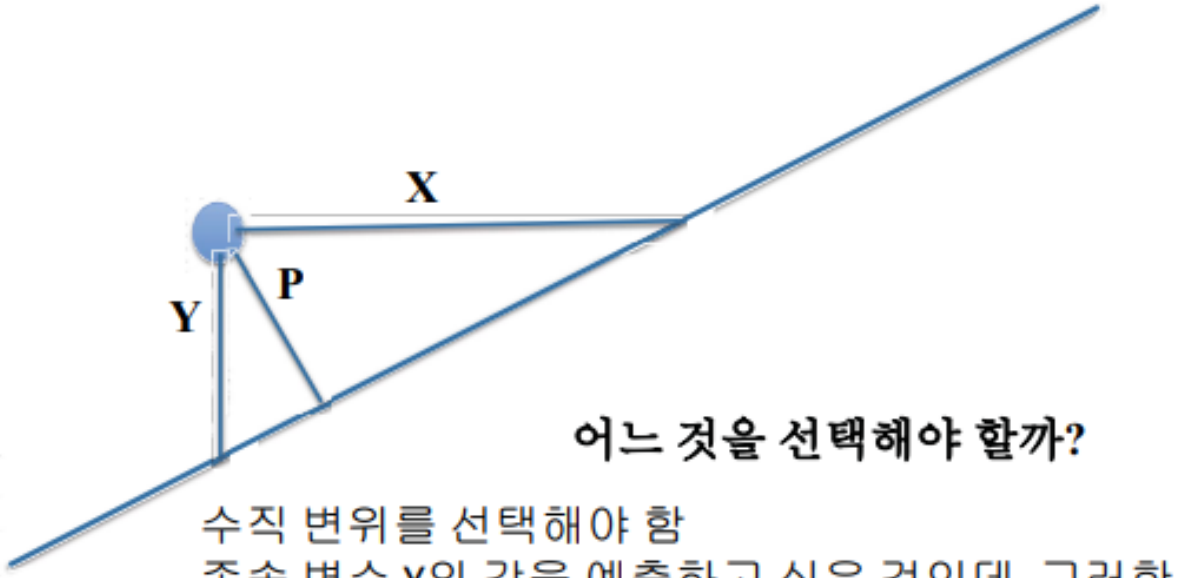
데이터에 곡선 피팅

- 목표: 측정 값에 잡음이 있더라도 데이터를 표현하는 선 찾기
- 데이터에 선을 맞추는 것(fit)은 독립 변수를 종속 변수의 추정값(Prediction)에 관련시키는 방법을 찾는 것
  - 독립변수 x축
  - 종속변수 y축
- 잘 맞는 직선이 있더라도 얼마나 쓸만 한지 알 수 있어야함
  - 목적 함수(Objective function) 필요



거리 측정

- 목적 함수 정의 후 최소화하는 최적의 선을 찾아야 함 (추세선)
- 3가지 방법이 존재
- x축을 따라 표현된 데이터들은 independent한 데이터이기 때문에 x축 거리 차이는 의미가 없음
- p도 말이 되는 것 같지만 이후 저 거리를 최소화해야 하는 경우를 보여줄 것



지금까지 정리

- **직선을 찾기 위해**
  - 측정값과 잡음을 고려
  - 독립 변수인 x축을 종속 변수 y축에 관련 시키는 방법
  - 직선이 있더라도 얼마나 적절한지 측정해야 함
- **목적 함수가 필요**
  - 얼마나 정확한지 측정
  - 목적 함수를 정의하고 나면 최소화하는 선을 찾음
  - 최적의 선, 목적 함수를 최소화하는 선을 찾음 -> 추세선(Best fit)
- 직선에서 측정값들의 거리의 합이 **최소화**가 되는 곳을 찾기
  - 종속 값을 예측하려고 함
  - 차이, 불확실성은 수직 변위임

Least Squares Objective Function

$$\sum_{i=0}^{len(observed)-1} (observed[i] - predicted[i])^2$$

- 선 하나를 가정
  - y = ax + b (a와 b 값을 알고 있을 경우)
- 추정값: 위의 식에 x값이 주어졌을 때, 저 직선이 y값을 예측하는 값
- 관찰값과 추정값 차이를 제곱
- 제곱
  - 부호가 사라짐 (관찰값이 추정값보다 작든 크든 상관이 없어짐)

Least Squares Objective Function

$$\sum_{i=0}^{len(observed)-1} (observed[i] - predicted[i])^2$$

- 정확히 분산은 아니지만 분산과 비슷함
- 해당 식은 분산과 관찰 횟수를 곱한 값과 같음
- 즉, 관찰 횟수로 나누면 분산이 됨
- 나눈 이후에 루트를 하면 표준편차가 됨
- 값들이 서로 얼마나 멀리 떨어져 있는지 알려주기 때문 중요

어떻게 하면 추세선을 찾을 수 있을까?

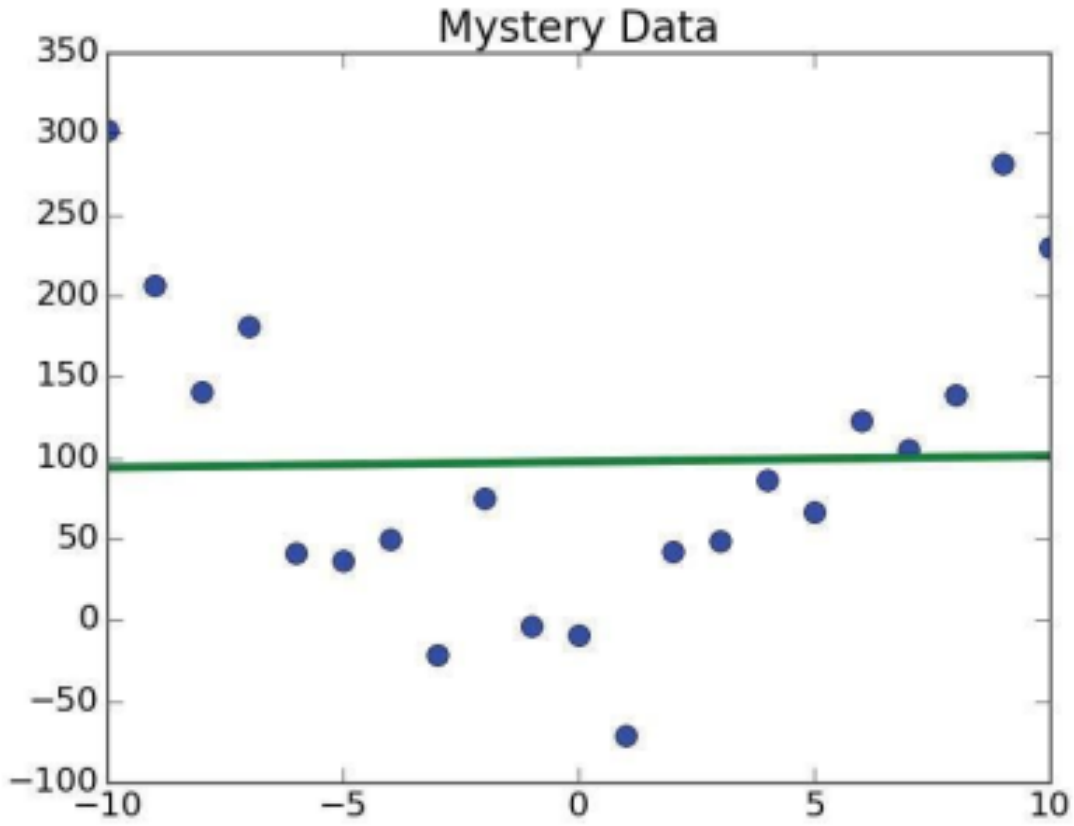
- 다항식은 c * x^p 의 형태를 가짐
- c는 상수, p는 지수, x는 구하고자 하는 자유 변수
- 다항식의 차수는 차수가 가장 큰 항의 지수
- 가정: y = ax + b에서 a와 b를 알고자 함
- 차 제곱의 합을 이용, 차 제곱의 합은 추세의 적합성을 나타냄
- 즉 차의 제곱이 최소화 되는 a와 b의 값을 구하는 것

Another Experiment

- 새로운 Dataset을 이용

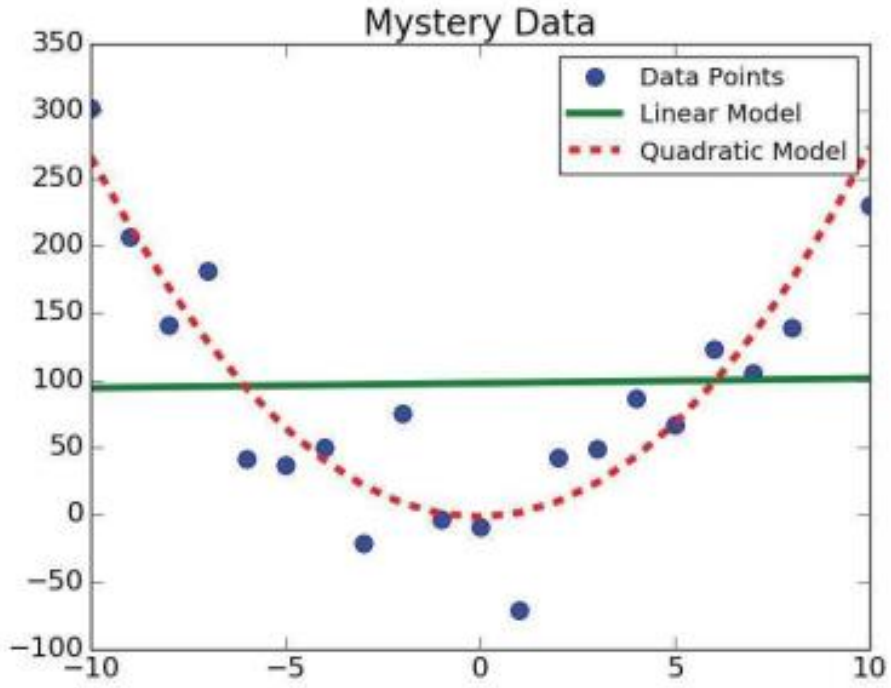
Another Experiment

- 직선 피팅을 이용
- 만족스럽지 않은 선
- 더 높은 차수를 이용해보자



추세선의 적합성을 파악하는 방법

- 2차 함수를 이용
- 뭐가 나은지 어떻게 결정하지?
- 그냥 서로 비교하는 방법(상대적인 관점)
- 절대적인 관점으로 보는 방법



상대적인 관점

- 어떤 곡선이 더 좋은 예측값(추정값)을 제공할까?
- 상대적인 관점으로 평균 제곱 오차를 최소화하는 곡선을 찾았으므로, 단순히 그 오차의 값을 통해 적합도를 평가할 수 있음

선형 모델의 평균 제곱 오차의 평균 = 9372.73078965
2차 모델의 평균 제곱 오차의 평균 = 1524.02044718

절대적인 관점

선형 모델의 평균 제곱 오차의 평균 = 9372.73078965
2차 모델의 평균 제곱 오차의 평균 = 1524.02044718

- 1524라는 값이 정말 좋은 값일까요?
- 이보다 더 좋은 게 있는지 어떻게 알 수 있을까요?
- 이 문제를 해결하기 위해 결정 계수(coefficient of determination), R^2 을 사용

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \mu)^2}$$

$\sum_i (y_i - p_i)^2$ ← 추정 오차

$\sum_i (y_i - \mu)^2$ ← 측정 데이터의 변이성

데이터 자체의 편차를 의미

Y_i 는 측정값
P_i 는 추정값
μ 는 측정값의 평균

절대적인 관점

$$R^2 = 1 - \frac{\sum_i (y_i - p_i)^2}{\sum_i (y_i - \mu)^2}$$

표본의 수로 나누면 분산 값을 얻을 수 있음

```
def rSquared(observed, predicted):
    error = ((predicted - observed)**2).sum()
    meanError = error/len(observed)
    return 1 - (meanError/numpy.var(observed))
```

- 오차(error)를 구한다
- meanError을 구한다
- meanError을 구하는 이유는 분모 계산을 쉽게 하기 위해서 구함

절대적인 관점

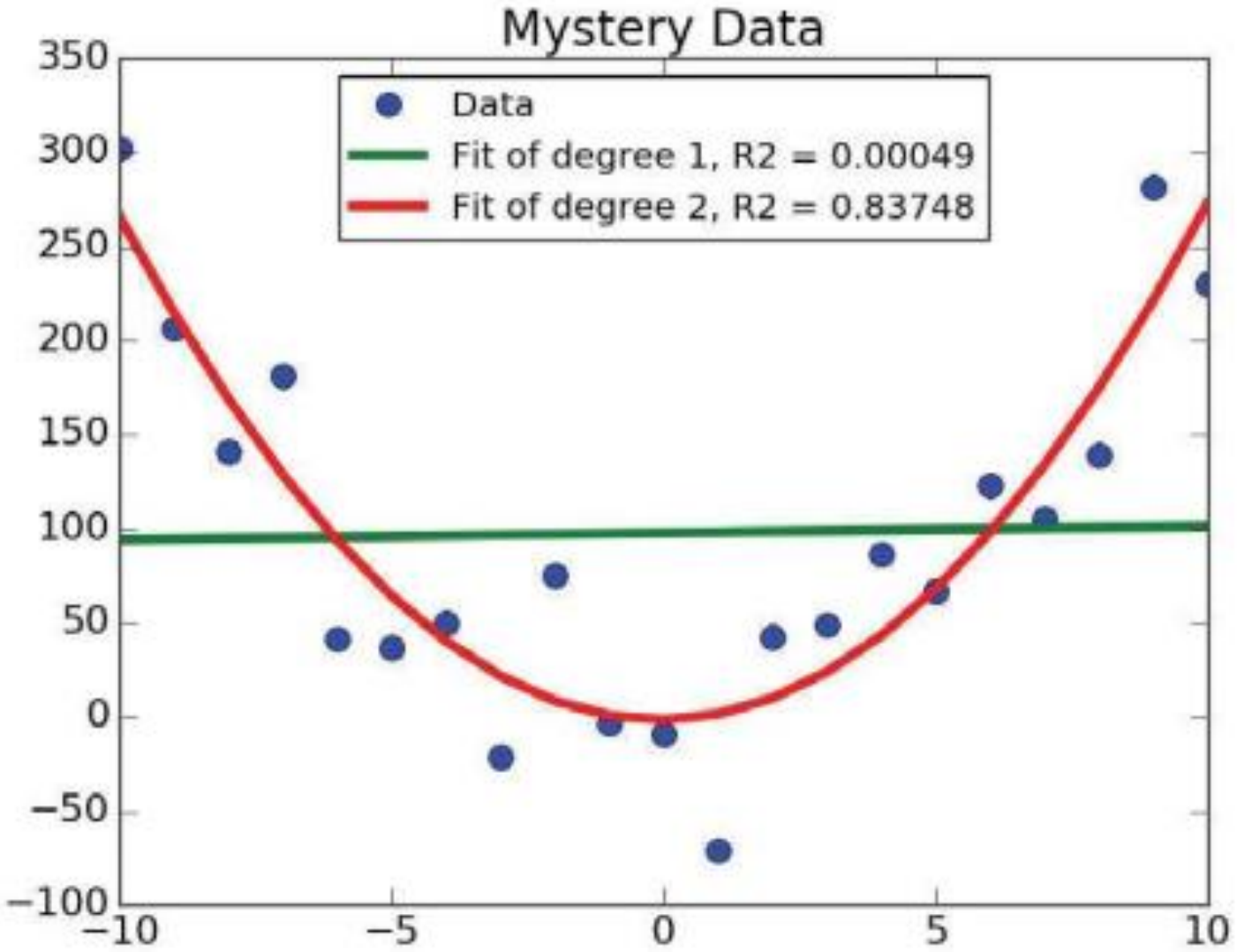
- R^2 = 1 이라면 매우 좋은 값
- 즉, 모든 데이터를 설명할 수 있다는 의미

- R^2 = 0 이라면 아무런 관계가 없다는 의미

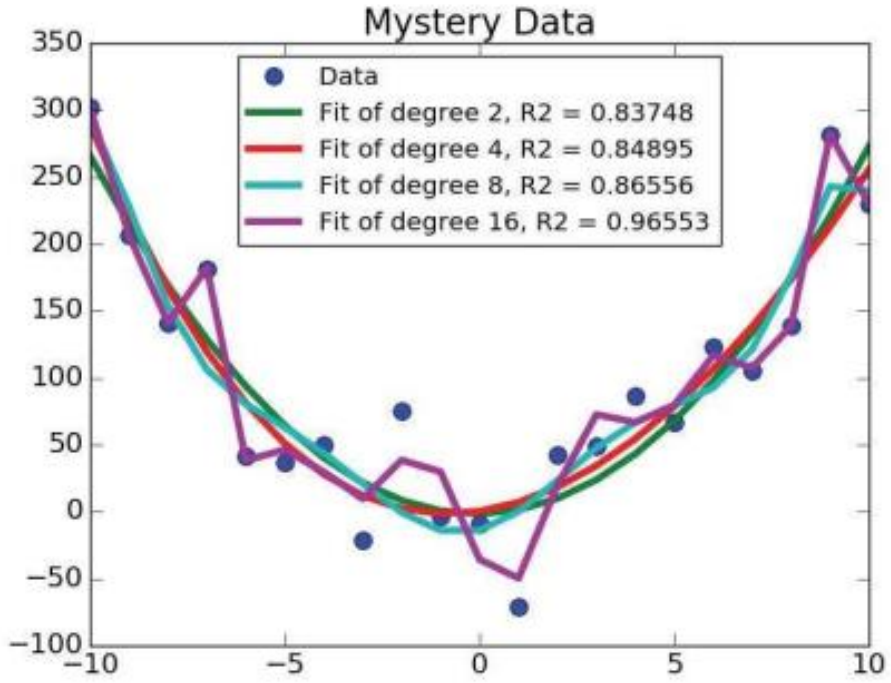
- R^2 = 0.5 이라면 전체의 0.5를 설명한다는 의미

- R^2 값이 1에 가까운 시스템을 구축하는 것이 목표

절대적인 관점

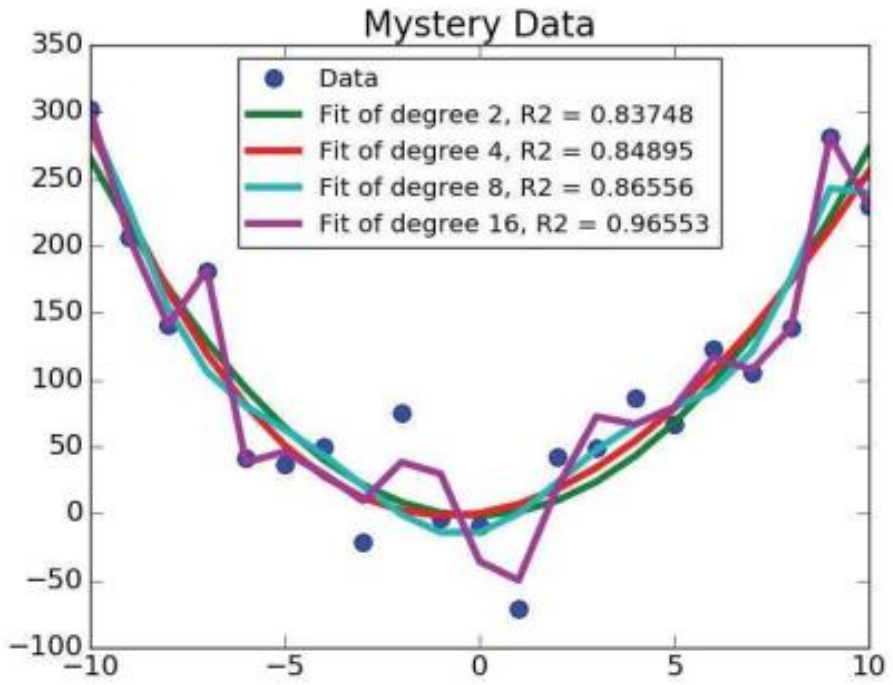

Mystery Data
Data
Fit of degree 1, R2 = 0.00049
Fit of degree 2, R2 = 0.83748
350
300
250
200
150
100
50
0
−50
−100
−10
−5
0
5
10

절대적인 관점



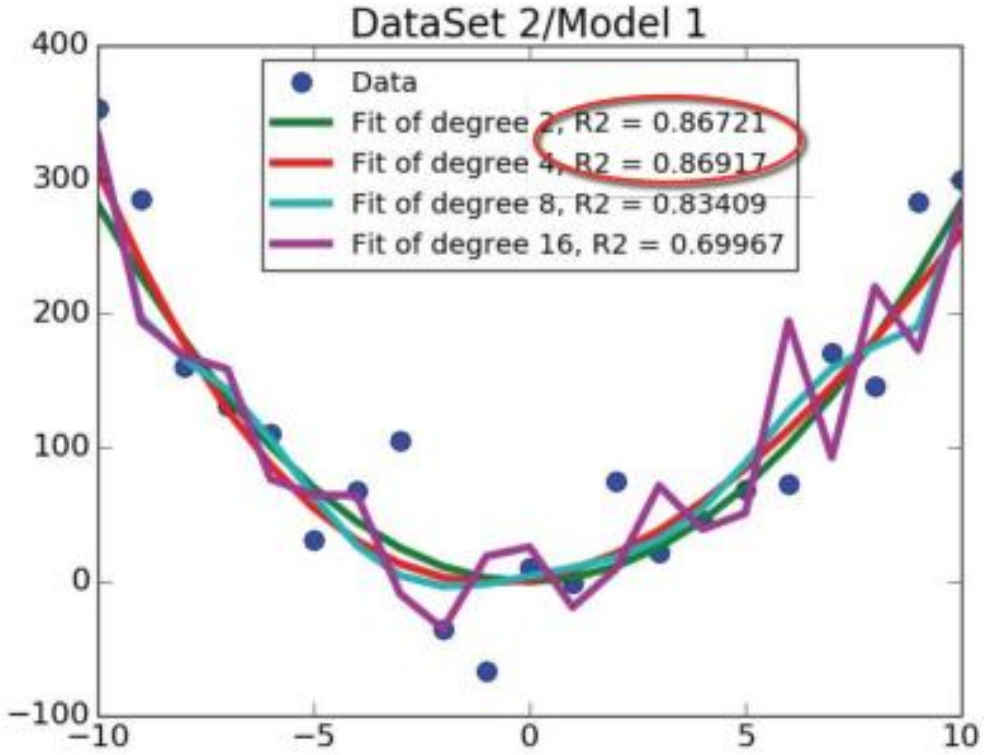
- 차수를 높여서 실험해봅시다.
- 16차식에서는 매우 좋은 값을 보여줍니다.
- 높은 R^2값을 갖는 다고 해서 16차식을 사용해야 하는 것은 아닙니다.

높은 차수의 모델

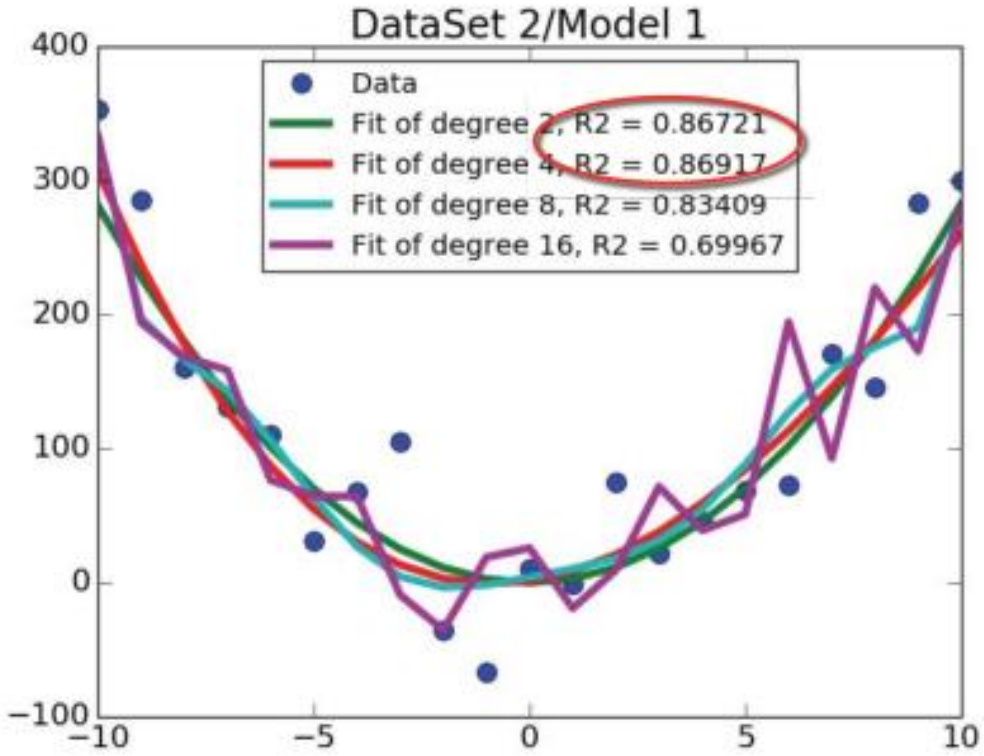


- 사실 이 해당 데이터들은 2차식에 random한 값들을 더한 데이터들입니다.
- 그런데 왜 16차식이 가장 적합하게 나올까요?
- 여기 오차는 "훈련 오차"입니다
- 해당 그림의 선들은 학습 데이터에 대한 표현

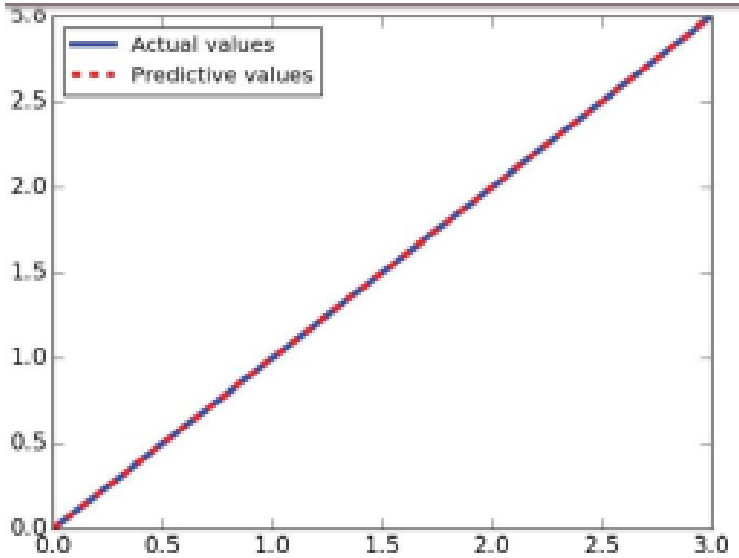
Overfitting



- 얼마나 잘 표현하는지 알기 위해 학습한 데이터 외에 다른 데이터를 이용하려고 함
- 2차 혹은 4차 식이 더 좋은 결과를 보여줌
- 이런 현상을 Overfitting이라고 부름
- 우리가 원하는 데이터 이외의 노이즈도 같이 포함되었기 때문에 발생



- 그렇다면 어떻게 더 일반적인 선을 그릴 수 있을까?
- 모델의 복잡도를 올리면서 훈련 데이터에 부적합한 추세선을 얻을 수 있을까? -> 못한다
- 더 높은 지수를 가지고 있고, 데이터에 노이즈가 많은 경우 모델은 노이즈까지 포함하여 추세선을 만들려고 합니다.



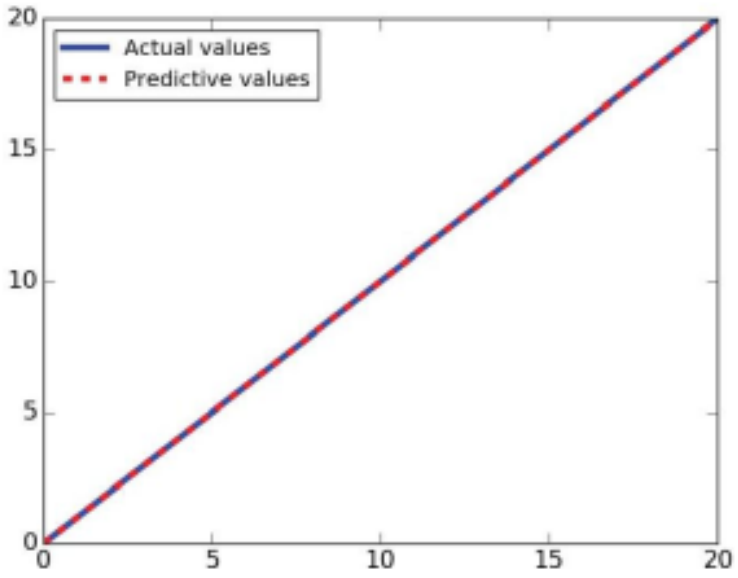
$y = ax^2 + bx + c$ 우리의 모델

$y = 0x^2 + 1x + 0$

$y = x$

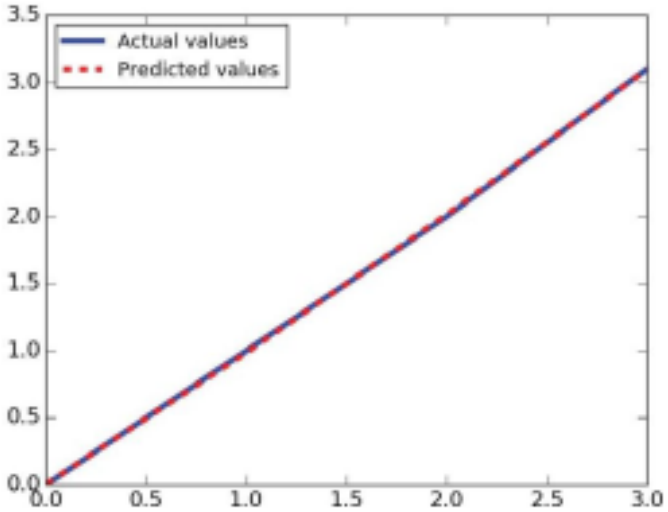
$R\text{-}squared = 1.0$

- 높은 차수가 필요하지 않음을 말하고 있음



R−squared = 1.0

- 새로운 데이터 20을 추가
- 같은 모델을 가지고 예측을 할 수 있음
- 이 또한 완벽하게 예측
- 완벽한 데이터를 갖고 있을 경우 높은 차수를 더한다고 문제가 되지 않음

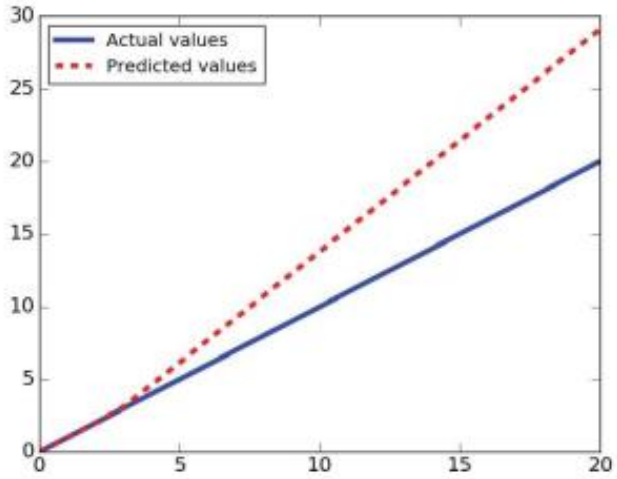


$$y = ax^2 + bx + c$$

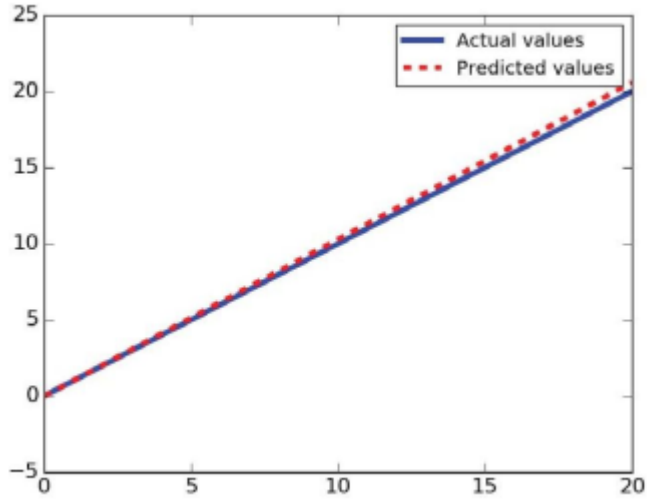
$$y = .025x^2 + .955x + .005$$

$$R\text{-squared} = 0.9994$$

- 작은 노이즈를 추가
- 아직까지 좋은 결과 값을 보여줌



R−squared = 0.7026



R−squared = 0.9988

- 새로운 데이터를 추가 (고의적으로 노이즈를 추가)
- 어떻게 하면 고칠 수 있을까?
- 차수가 높은 모델을 그리는 것의 단점은 무엇일까?
- 오른쪽 그림은 1차식으로 표현
- 알맞은 차수의 모델을 그린다면 노이즈는 크게 문제 되지 않음

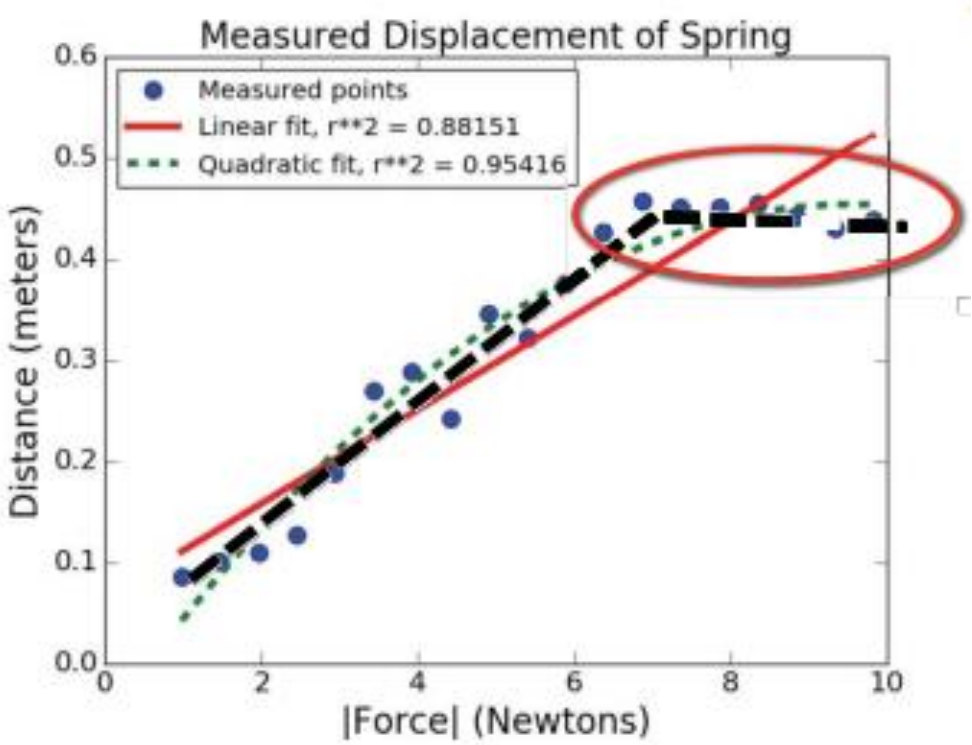
지금까지 정리

- 충분히 복잡하지 않은 모델은 데이터를 잘 설명하지 못함
- 너무 복잡한 모델은 훈련 데이터에 Overfitting 함
- 이것들의 중간 점을 찾는 것이 목표
- 모델이 간단하면서도 데이터를 잘 설명할 수 있는 중간점에 있는 모델

그럼 어떻게 알맞은 모델을 찾을 수 있을까?

1. 낮은 차수 모델로 시작
2. 새로운 데이터에도 잘 맞는지 확인
3. 차수를 높임

- 찾을 때까지 1~3 과정을 반복
- 정확도가 다시 떨어지기 시작할 때가 적합한 모델을 찾았다는 신호



- 훅 법칙에 관한 실험 데이터
- 2차 곡선이 더 잘 맞음
- 훅 법칙은 1차 직선 형태를 가짐
- 제한: 용수철의 탄성 한계에 도달하기전까지 성립
- 데이터를 2개 부분으로 나누어서 선을 찾을 수 있음

확실한 이론이 없다면...

- 이론을 모르고 있다고 해도 적합한 추세선을 찾을 수 있을까?
- 교차 검증 결과를 기반으로 모델의 복잡도 선택
- Dataset이 작다면 **Leave one out cross validation(LOOCV)**을 사용
- Dataset이 충분히 크면, **k-fold 교차 검증** 또는 **반복 무작위 추출 검증** 이용

Leave one out cross validation(LOOCV)

```
Let D be the original data set

testResults = []
for i in range(len(D)):
    training = D[:].pop(i)
    model = buildModel(training)
    testResults.append(test(model, D[i]))

Average testResults
```

반복 무작위 추출

```
Let D be the original data set
    n be the number of random samples
        usually n between 20% and 50%
    k be number of trials

testResults = []
for i in range(k)
    randomly select n elements for testSet,
        keep rest for training
    model = buildModel(training)
    testResults.append(test(model, testSet))

Average testResults
```

왜 여러 개의 데이터세트로 검증해야 할까?

Here are the R^2 values for each trial of linear fit

- [0.7828002156420516, 0.8063796402505206 7,
  0.7963713275727426 5, 0.7843388574321190 6,
  0.7600111202485312 4, 0.5708893650703574 8,
  0.7211540856258902 3, 0.7435827676214902 3,
  0.7903145537514850 7, 0.7792023858639947 1]

- 만약 한 번만 실행했더라면 동그라미 친 값만 얻을 수 있고 매우 낮은 R^2값을 얻고 linear가 나쁘다고 생각할 수 있음
- 만약 한 번의 분할만 실행하여 이러한 결과를 얻었다면, 선형 모델의 유효성에 대해 다른 결론에 도달했을 것