

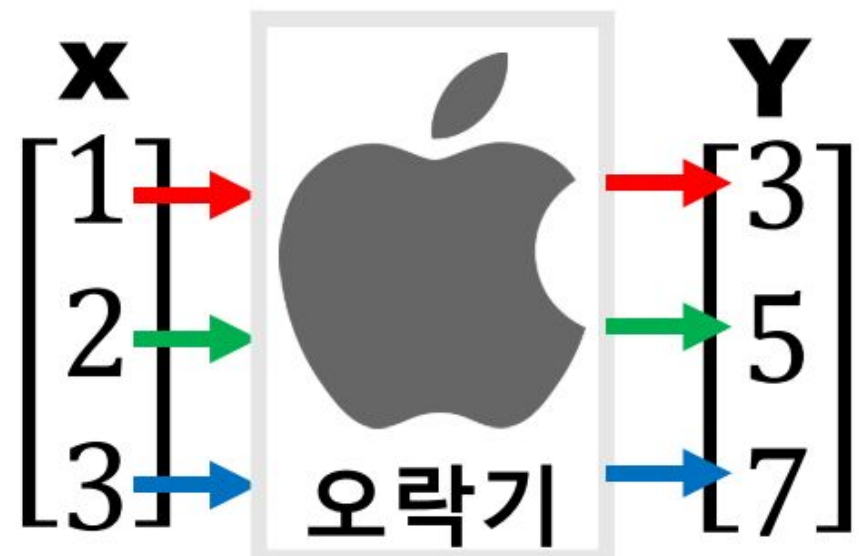
Machine learning

Overfitting 극복 & Linear Regression 2

머신러닝의 학습 방법들

- **Probability theory-based learning**
- **Gradient descent-based learning**
- **Information theory-based learning**
- **Distance similarity-based learning**

목표 : 오락기 구조를 아는 것
 이유 : 새버튼 '0'의 결과 예측!



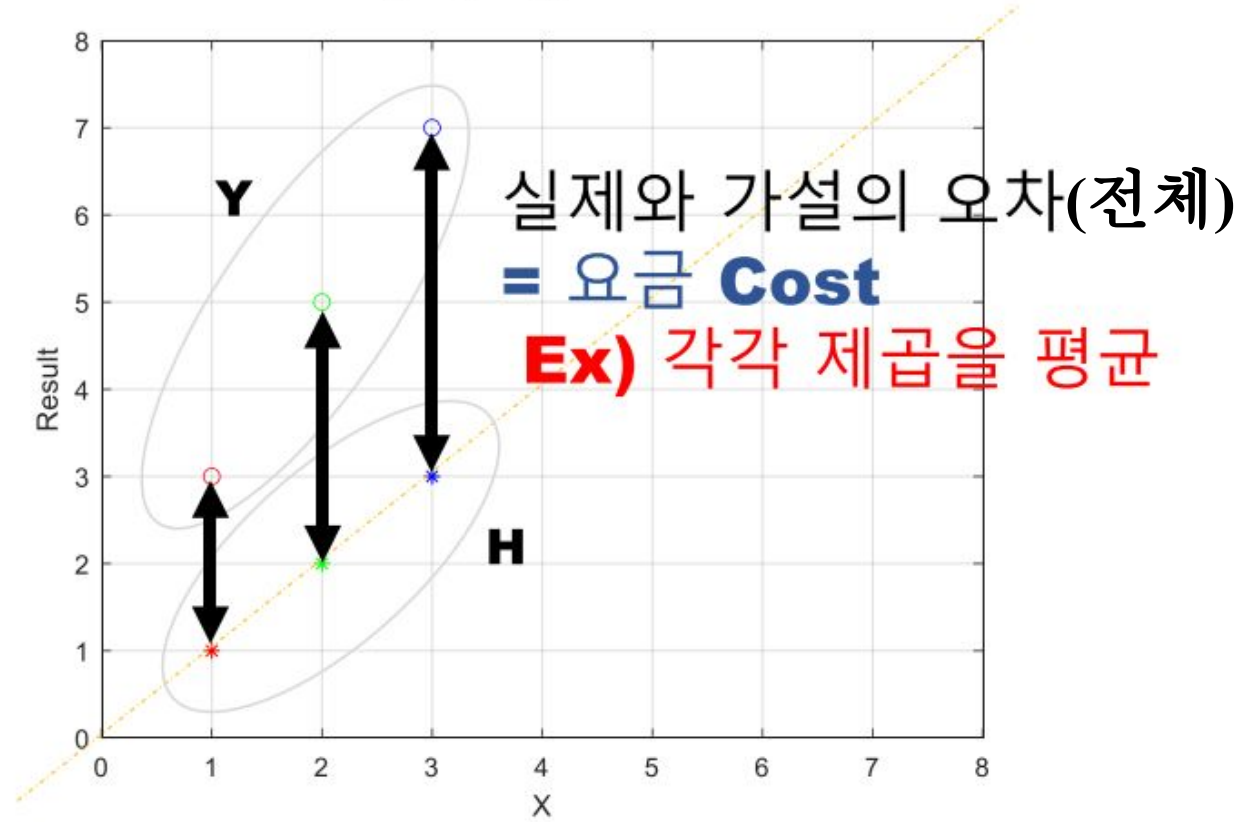
실제, $y_i = @x_i + \$$

가설, $H(w, b) = Wx_i + b$

$i = 1, 2, 3$ 까지는 경험
 $i = 4$ 는 새로운 입력

가설, $Hypothesis(W, b)$
 $= Weight \cdot x_i + bias$

초기값 $W = 1, b = 0$ 으로 가정하면,
 $H(1, 0) = 1 \cdot X + 0$

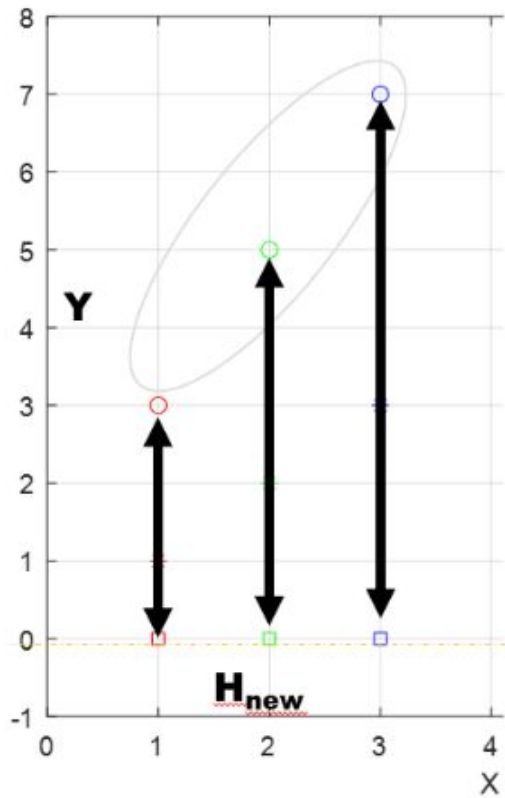
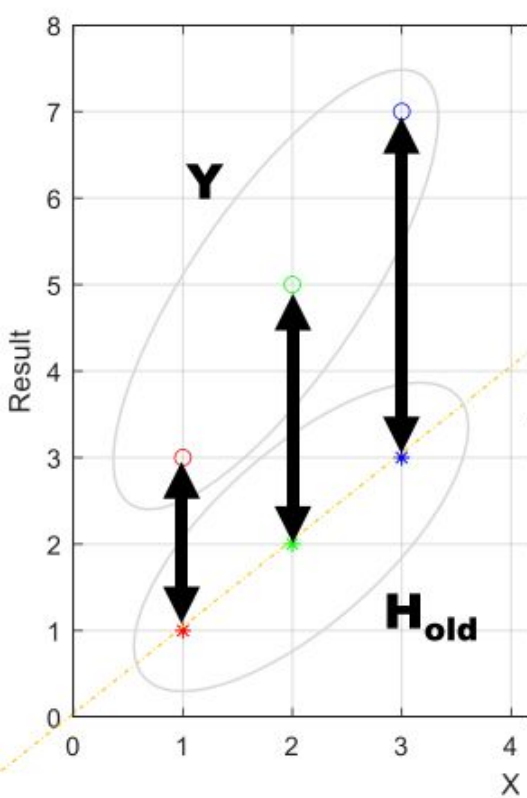


초기값 $W = 1, b = 0$ 으로 가정하면, $\rightarrow W = 0, b = 0$ 으로 바꾸면

$$H(1,0) = 1 \cdot X + 0$$

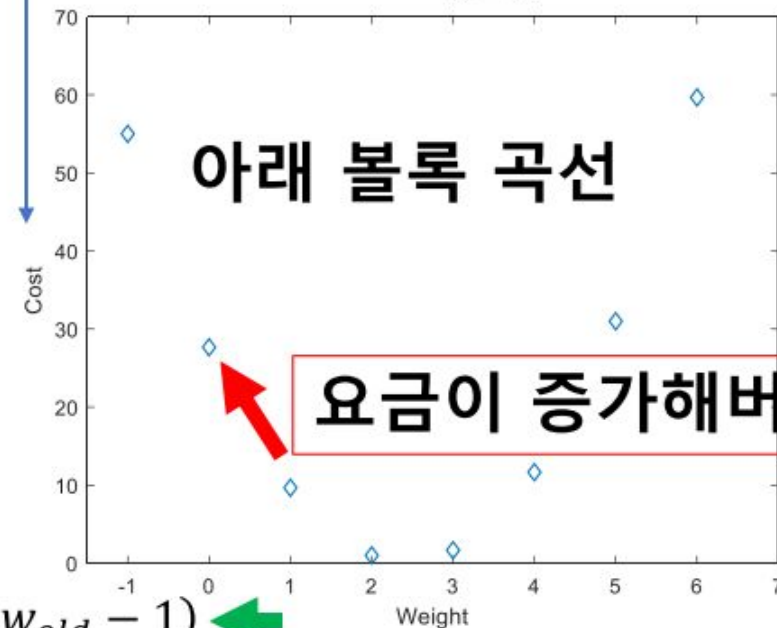
$$H(0,0) = 0 \cdot X + 0$$

$$cost(1,0) = \frac{2^2 + 3^2 + 4^2}{3} = \frac{29}{3} \rightarrow cost(0,0) = \frac{3^2 + 5^2 + 7^2}{3} = \frac{83}{3} (>2.8배)$$



일반화

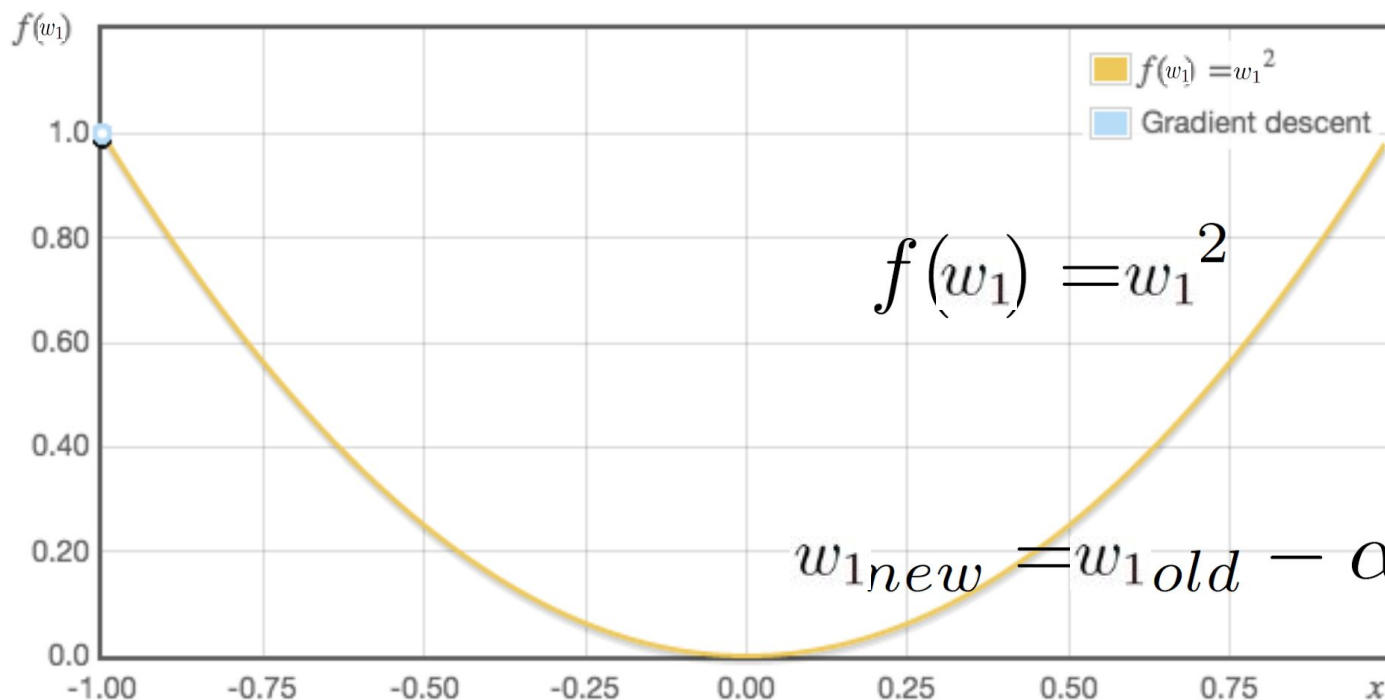
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m \{(Wx_i + b) - y_i\}^2$$



$$(w_{new} = w_{old} - 1)$$

Linear regression with GD

만약, 오른쪽 공식의 cost 함수를
w1만의 이차 함수로 가정하면! 아래처럼,



$$\text{cost}(\mathbf{W}, \mathbf{b}) = \frac{1}{m} \sum_{i=1}^m \{(Wx_i + b) - y_i\}^2$$

$$\mathbf{W} := \mathbf{W} - \alpha \frac{\partial}{\partial \mathbf{W}} \text{cost}(\mathbf{W})$$

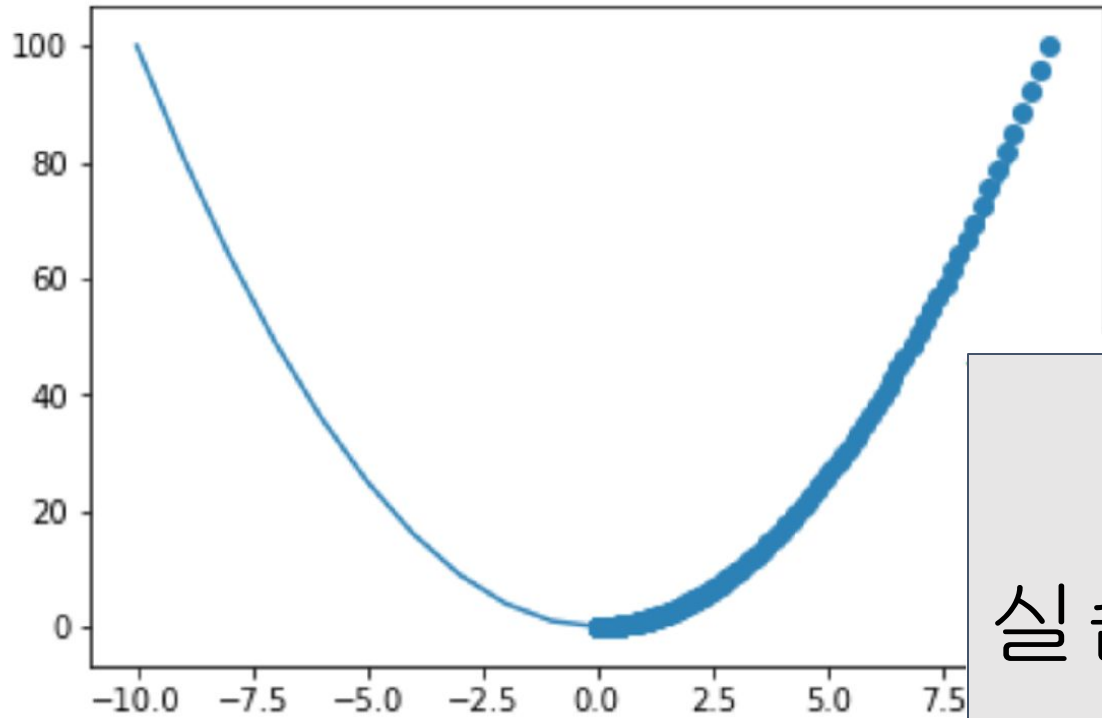
$$\mathbf{W} := \mathbf{W} - \alpha \frac{\partial}{\partial \mathbf{W}} \frac{1}{2m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})^2$$

$$\mathbf{W} := \mathbf{W} - \alpha \frac{1}{2m} \sum_{i=1}^m 2(Wx^{(i)} - y^{(i)})x^{(i)}$$

$$\mathbf{W} := \mathbf{W} - \alpha \frac{1}{m} \sum_{i=1}^m (Wx^{(i)} - y^{(i)})x^{(i)}$$

주의할 점: 목표는 Cost가 최소가 되는 것

GD를 이용하여 최소값 찾기



실습 영상

그전에! 필수 파이썬 Tip

[1] **append()**는 object를 맨 뒤에 추가합니다.

```
x = [1, 2, 3]
```

```
x.append([4, 5])
```

```
print(x)
```

```
[1, 2, 3, [4, 5]]
```

[2] **extend()**는 iterable 객체(리스트, 튜플, 딕셔너리 등)의 엘리먼트를 list에 appending시킵니다.

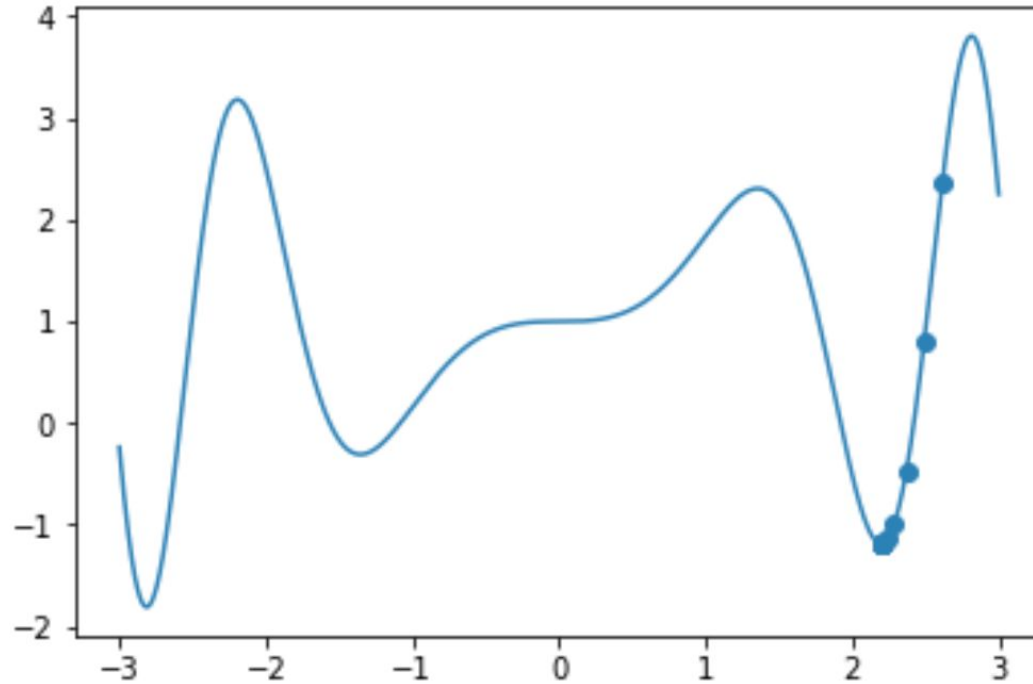
```
x = [1, 2, 3]
```

```
x.extend([4, 5])
```

```
print(x)
```

```
[1, 2, 3, 4, 5]
```

굴곡이 많은 경우

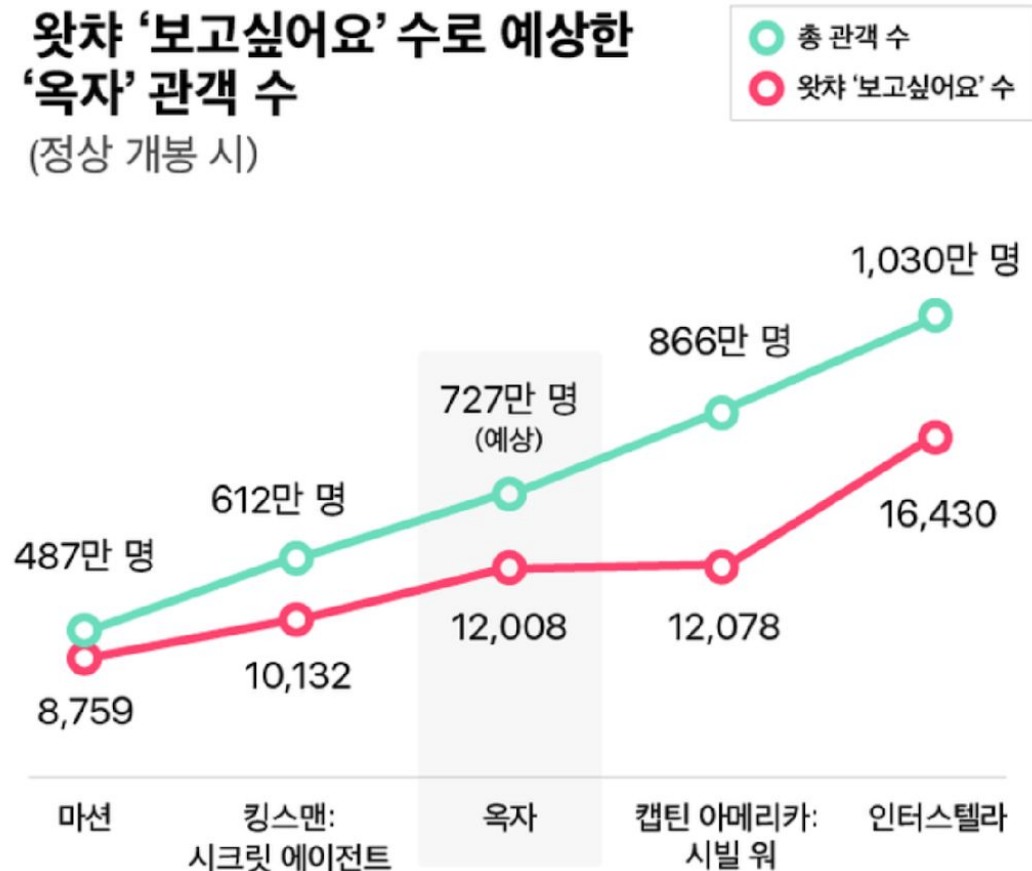


Local minimum

시작점에 따라 다른 최소값을 찾는다.

왓차 '보고싶어요' 수로 예상한 '옥자' 관객 수

(정상 개봉 시)



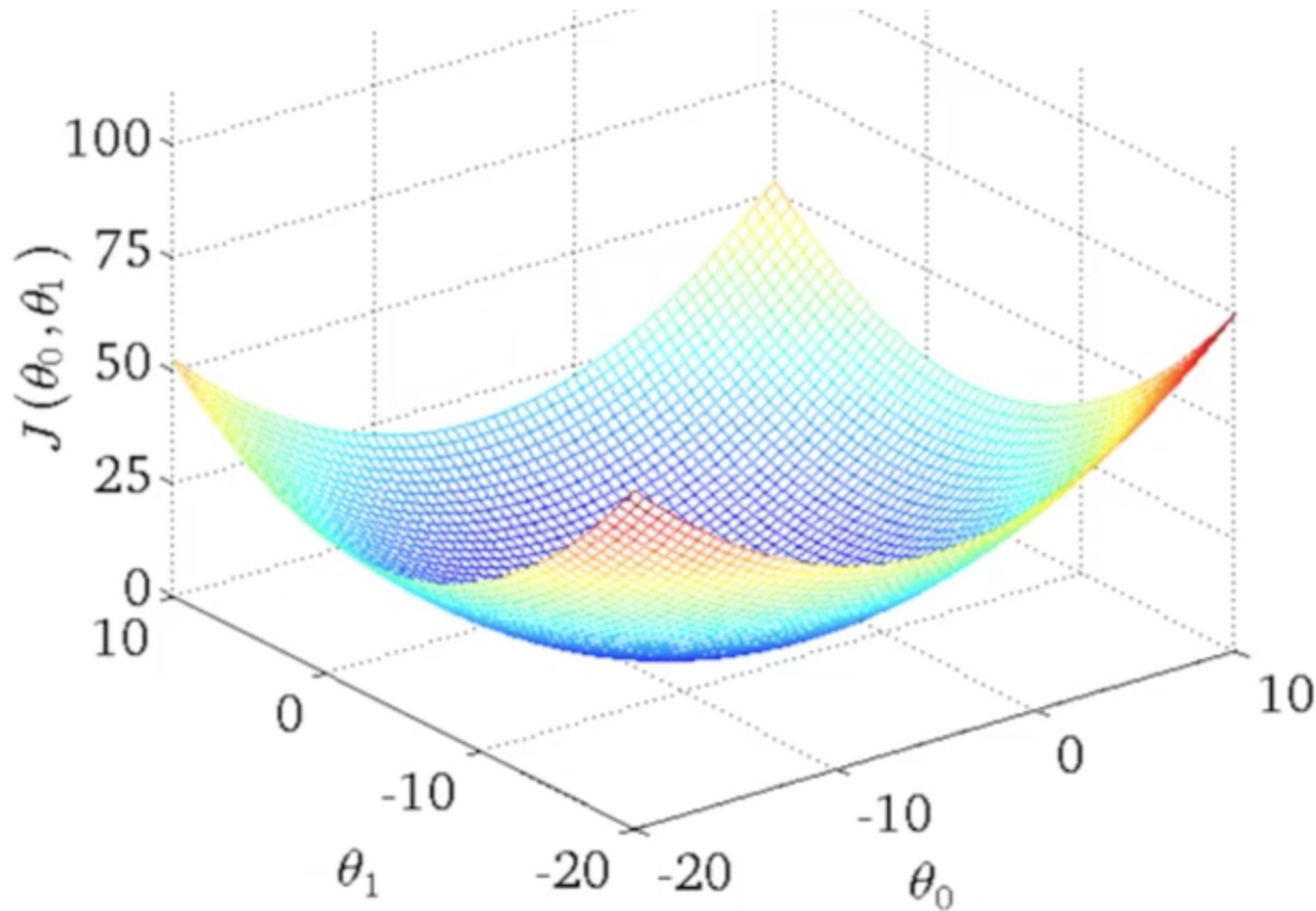
$$\mathbf{y} = \begin{bmatrix} 487 \\ 612 \\ 866 \\ 1030 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 & 8,759 \\ 1 & 10,132 \\ 1 & 12,078 \\ 1 & 16,430 \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

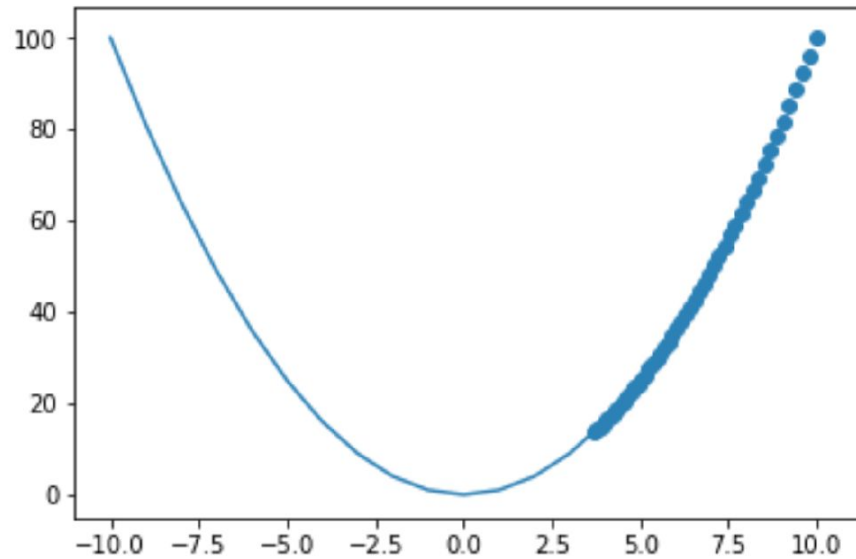
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^m (w_1 x^{(i)} + w_0 - y^{(i)})^2$$

Minimize $J(w_0, w_1)$

얼만큼씩 변화를 주며 훈련시킬 것인가

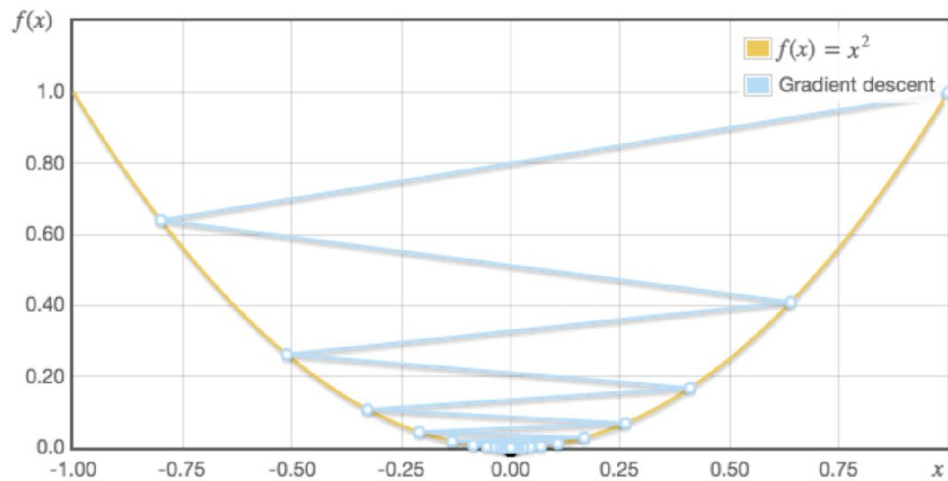


Learning rate가 너무 작을 경우



끝까지 못감
시간이 오래 걸림

Learning rate가 너무 클 경우



데이터가 튀는 문제가 생김
수렴하지 못하는 경우 생김

GD를 이용한 학습

결국 파라미터의 업데이트

Linear regression with GD

- Learning rate, Iteration 횟수 등이
Parameter 설정에 영향을 주므로, Hyperparameter로 불림.
- 설계에 따라 최적값에 수렴하지 못할 수도 있음

1. 명사 (일괄적으로 처리되는) 집단[무리]
2. 명사 한 회분(한 번에 만들어 내는 음식기계 등의 양)
3. 동사 (일괄 처리를 위해) 함께 묶다

Full-batch gradient descent

- 안정적인 Cost 함수 수렴 (넓게 보므로)
- 지역 최적화 (Local Optimum) 가능성 존재
- 메모리 문제 (ex – 30억개의 데이터를 한번에?)
- 대규모 dataset -> 모델/파라미터 업데이트가 느려짐



stochastic 미국·영국 [stəkæstik]

추계학(推計學)의, 확률(론)적인

Stochastic gradient descent

Stochastic gradient descent

- 원래 의미는 dataset에서 random하게 training sample을 뽑은 후 학습할 때 사용함

shuffle 미국·영국 ['ʃʌfl]  영국식  ★

1. 발을 (질질) 끌며 걷다
2. (어색하거나 당황해서 발을) 이리저리 움직이다
3. (게임을 하기 위해 카드를) 섞다

- Data를 넣기 전에 Shuffle

1: procedure SGD

2: shuffle(X)

▷ Randomly shuffle data

3: for i in number of X do

4: $\theta_j := \theta_j - \alpha(\hat{y}^{(i)} - y^{(i)})x_j^{(i)}$

▷ Only one example

5: end for

6: end procedure

Stochastic gradient descent

‘지점’을 말할 때 mum과 ma를 씀

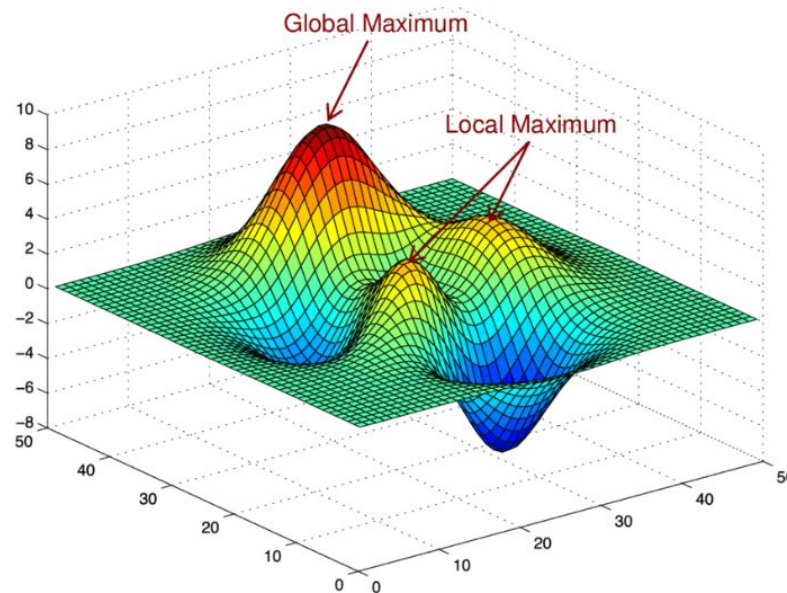
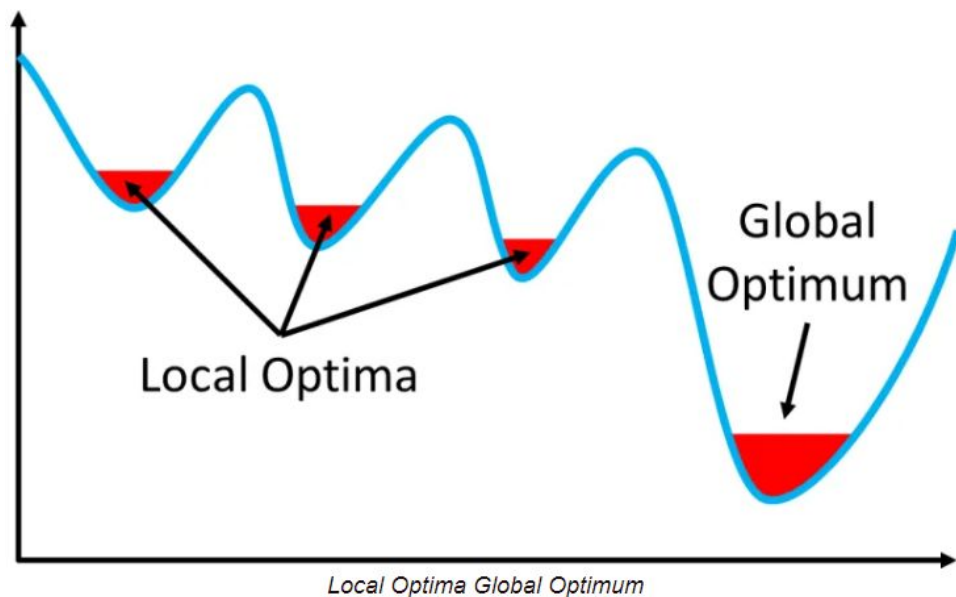
1. 지역 최적화(Local Optimum) 회피

2. 일부 문제에 대해 더 빨리 수렴

3. 대용량 데이터시 시간이 오래걸림 (전체 중에 섞는 것이므로)

Optimum의 복수형은
Optima


좋은건 최대 Maximum
나쁜건 최소 Minimum





Mini-batch (stochastic) gradient descent

Mini-batch SGD

- 일부 Mini 량의 데이터를 학습 + Shuffle
- 일부 Batch로 나눠서 하는 GD인데 SGD인 기법
- 가장 일반적으로 많이 쓰이는 기법

epoch 미국식 ['epək]  영국식 ['i:pək]

1. (중요한 사건·변화들이 일어난) 시대

iteration 미국·영국 [ˌɪtə'reɪʃn]  영국식 

1. 명사 (계산·컴퓨터 처리 절차의) 반복

epoch, batch-size, iteration

batch 미국·영국 [bætʃ]  영국식  +

1. 명사 (일괄적으로 처리되는) 집단[무리]

2. 명사 한 회분(한 번에 만들어 내는 음식기계 등의 양)

3. 동사 (일괄 처리를 위해) 함께 묶다

- Training 시기 (1차 시기, 2차 시기,..., n차 시기)
전체 Full-batch를 n epoch 관찰했다고 표현
- Batch-size라는 용어:
한 iteration 에 학습되는 데이터의 개수.
이것이 k-iteration 반복하여 한 epoch 완성
- 총 5,120개의 Training data에 512 batch-size라면
몇 iteration을 해야 1 epoch이 되는가?

epoch & mini-batch

So, after creating the mini-batches of fixed size,
we do the following steps in one epoch:

1. Pick a **mini-batch**.
2. Feed it to Neural Network (AI 방법, 한 종류).
3. Calculate the mean gradient of the **mini-batch**.
4. Use the mean gradient we calculated in **step** 3 to update the weights.
5. Repeat **steps** 1–4 for the **mini-batches** we created.

epoch & mini-batch

So, after creating the mini-batches of fixed size,
we do the following steps in one epoch:

1. Pick **a** mini-batch.
2. Feed it to Neural Network (AI 방법, 한 종류).
3. Calculate the mean gradient of the **mini-batch**.
4. Use the mean gradient we calculated in **step** 3 to update the weights.
5. Repeat **steps** 1–4 for the **mini-batches** we created.

Mini-batch SGD

1: **procedure** MINI-BATCH SGD

2: **shuffle**(X)

▷ Randomly shuffle data

3: $BS \leftarrow \text{BATCH SIZE}$

4: $NB \leftarrow \text{Number of Batches}$

5: $NB \leftarrow \text{len}(X) // BS$

6: **for** i **in** NB **do**

7: $\theta_j := \theta_j - \alpha \sum_{k=i \times BS}^{(i+1) \times BS} (\hat{y}^{(k)} - y^{(k)}) x_j^{(k)}$

▷ Batch-sized examples

SGD를 구현할 때
생각해 봐야 할 일들

그전에 잠깐 파이썬 numpy 팁!

Numpy에서 배열(ndarray) 복사 3가지

1. `b = a` [ndarray의 데이터와 속성을 모두 공유]

그저, b라는 이름 하나 더 생김 = b를 바꾸면 a에도 영향.

2. `b = a.view()` [보이는 뷰는 **안 변하게** 가져 온다!]

`b.shape=(1,15)`은 a 영향 없음

`b[0]=0`하면 a 영향 있음

3. `b = a.copy()` [완전히 새로운 변수로 복사됨]

Mini-Batch SGD

Operator	Description	Example
+	더하기	$a + b = 30$
-	빼기	$a - b = -10$
*	곱하기	$a * b = 200$
/	나누기	$b / a = 2.0$
%	나머지	$b \% a = 0$
**	제곱	$a ** c = 1000$
//	몫	$a // c = 3$

```
for epoch in range(epochs):
    X_copy = np.copy(X)
    if is_SGD:                                SGD 여부 → SGD일 경우 shuffle
        np.random.shuffle(X_copy)
    batch = len(X_copy) // BATCH_SIZE          한번에 처리하는 BATCH_SIZE
    for batch_count in range(batch):
        X_batch = np.copy(
            X_copy[batch_count*BATCH_SIZE : (batch_count+1)*BATCH_SIZE])
        # Do weight Update                    BATCH_SIZE 크기 만큼 X_batch 생성
    print("Number of epoch : %d" % epoch)
```

**Learning rate은
일정해야 하는가?**

Learning-rate decay

- 일정한 주기로 Learning rate을 감소시키는 방법
- 특정 epoch, t 마다 Learning rate를 감소
- Hyper-parameter 설정의 어려움

[1] 지수감소 $\alpha = \alpha_0 e^{-kt}$

[2] 1/t감소 $\alpha = \frac{\alpha_0}{(1 + kt)}$

종료조건 설정

- SGD과정에서 특정 값이하로 cost function이 줄어들지 않을 경우 GD를 멈추는 방법
- 성능이 좋아지지 않는/필요없는 연산을 방지함
- 종료조건을 설정
$$\text{epoch_end_condition} > \text{cost} - \text{previous_cost}$$
- epoch_end_condition도 hyperparameter.

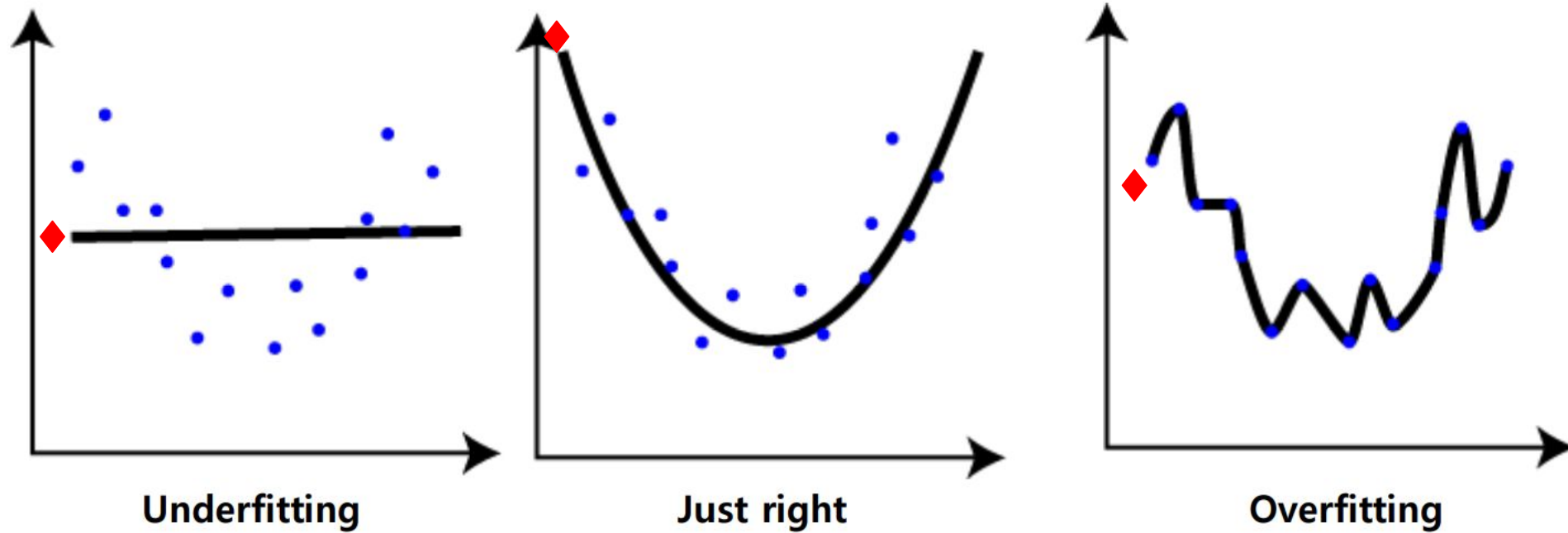
Overfitting

과적합

(훈련값에) 너무 딱 맞추다

Overfitting

- 학습데이터 과다 최적화 -> 새로운 데이터의 예측 정확도 감소



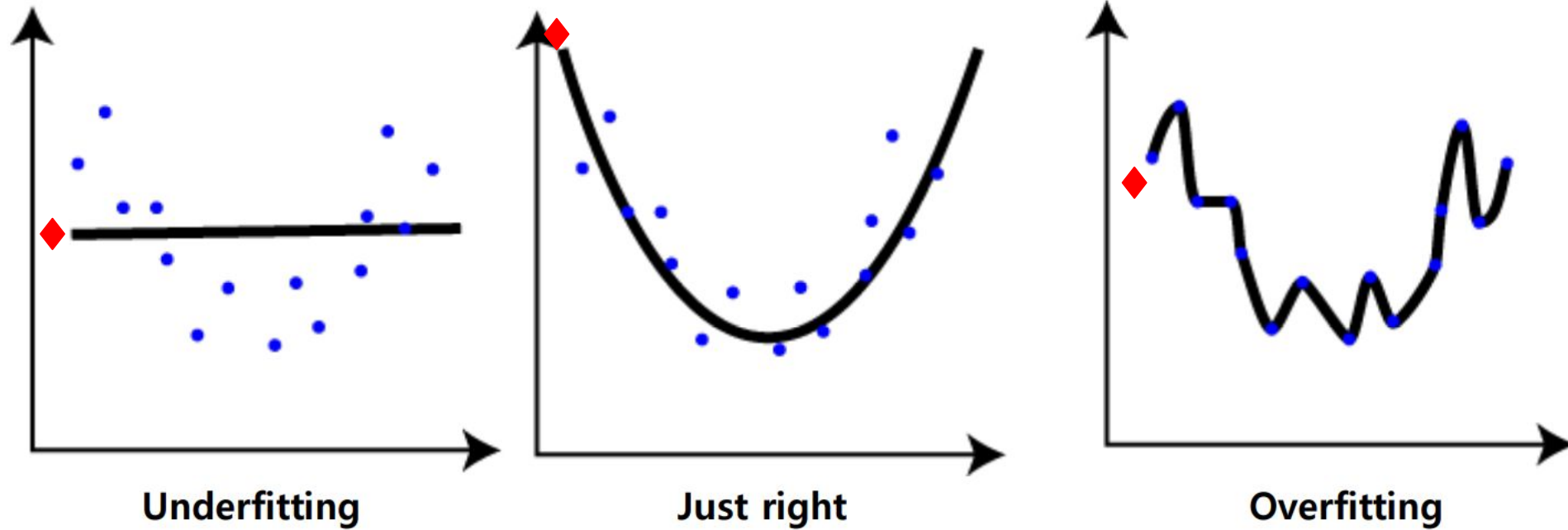
Overfitting

Occam's razor 오컴의 면도날

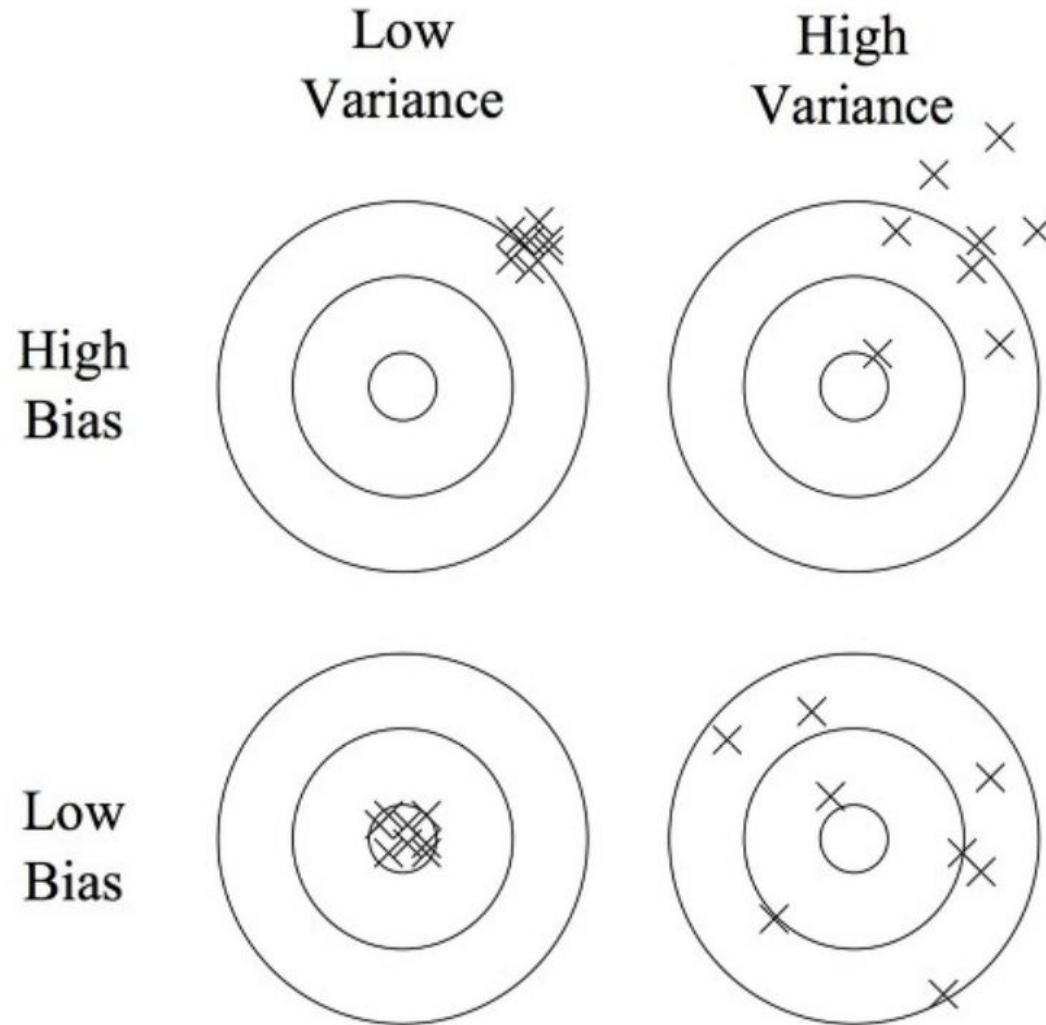
보다 적은 수의 논리로 설명이 가능한 경우,
굳이 많은 수의 논리를 세우지 말라

Overfitting

Bias-Variance tradeoff



Bias-Variance tradeoff



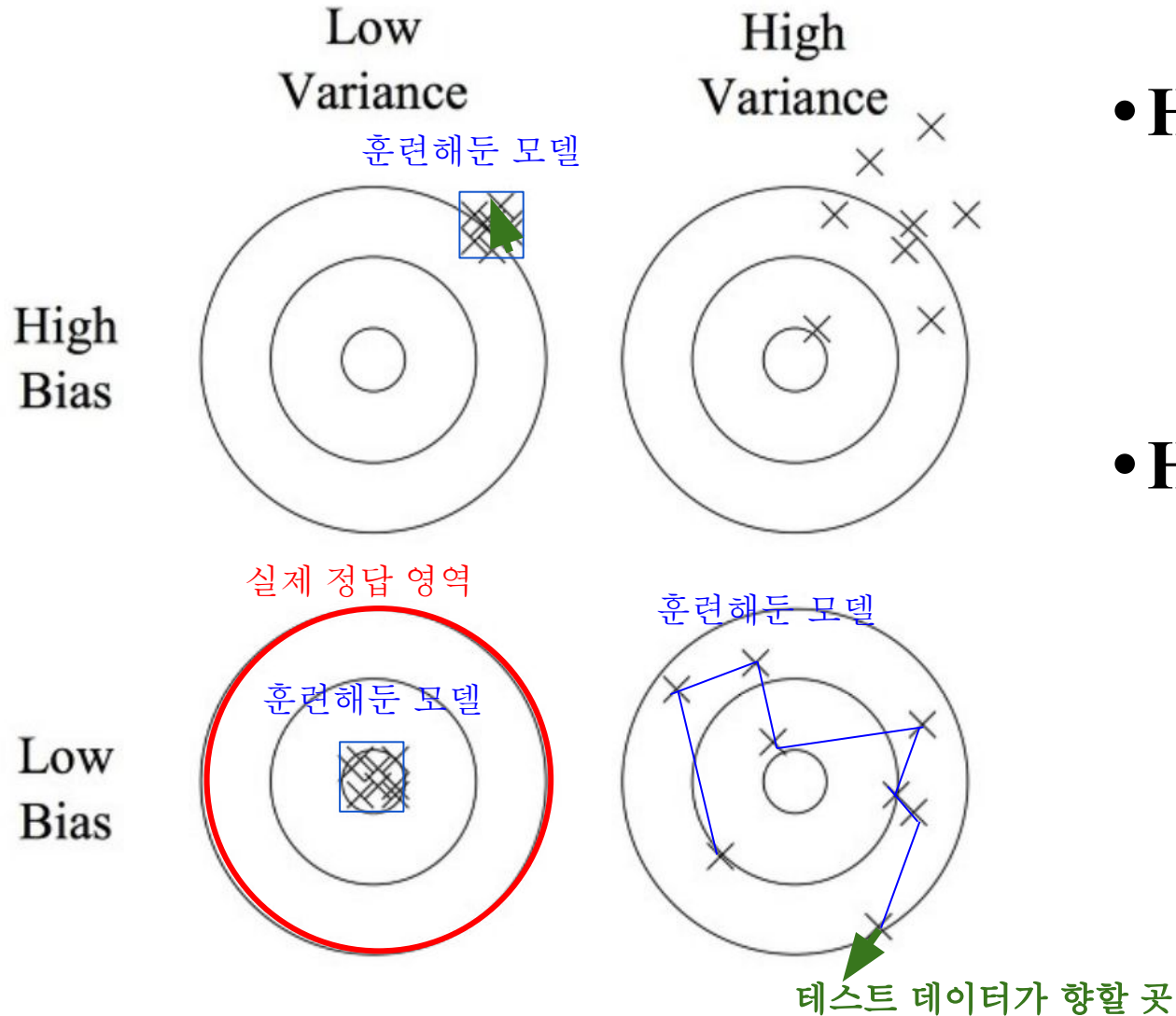
- **High bias ~ Underfitting**

- 원래 모델에 많이 떨어짐
- 잘못된 데이터만 계속 학습함

- **High variance ~ Overfitting**

- 모든 데이터에 민감하게 학습
- Error 고려가 적음

Bias-Variance tradeoff



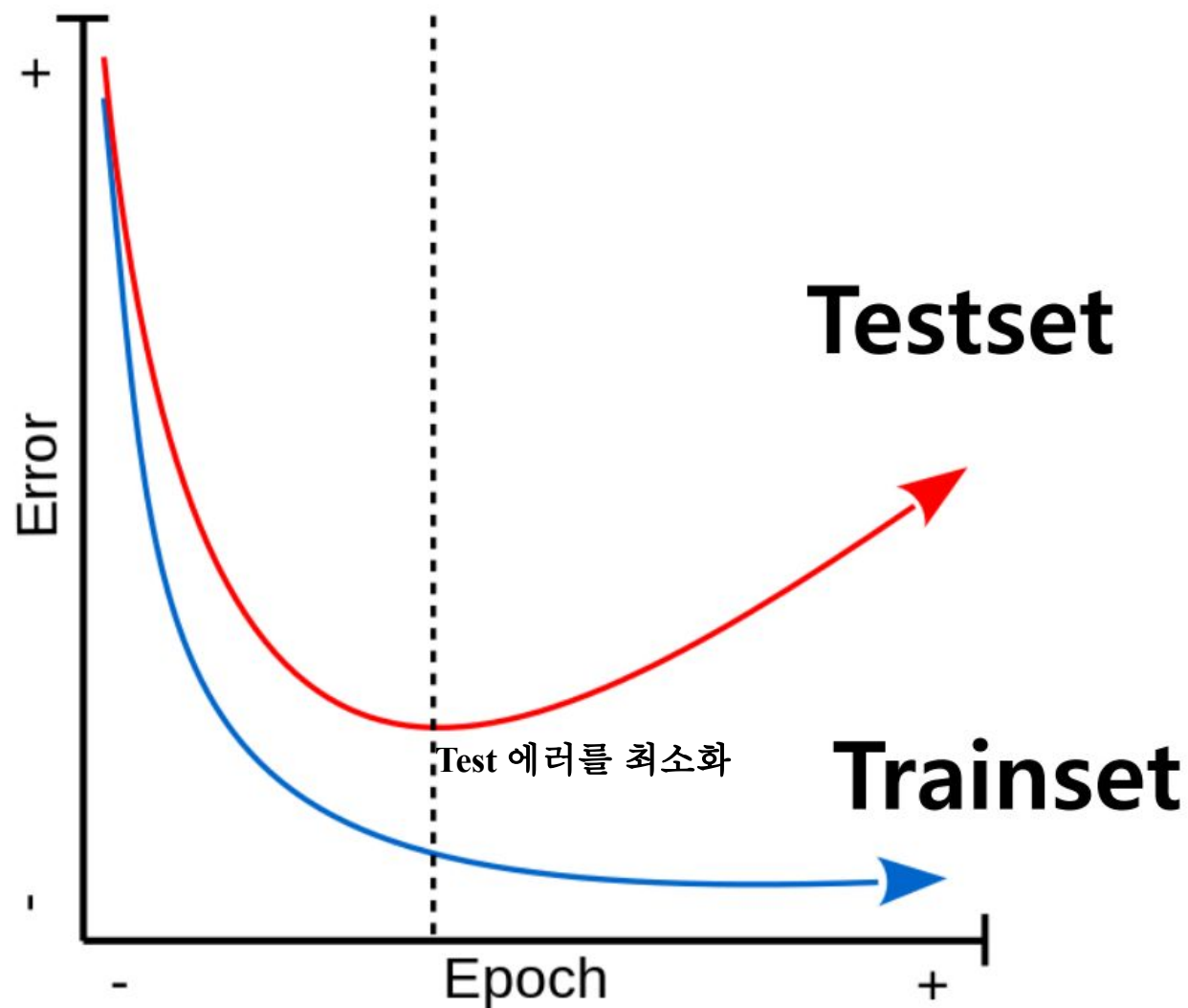
- **High bias ~ Underfitting**

- 원래 모델이 정답과 많이 떨어짐
- 잘못된 데이터만 계속 학습함

- **High variance ~ Overfitting**

- 모든 데이터에 민감하게 학습
- Error 고려가 적음

Training Error vs Test Error

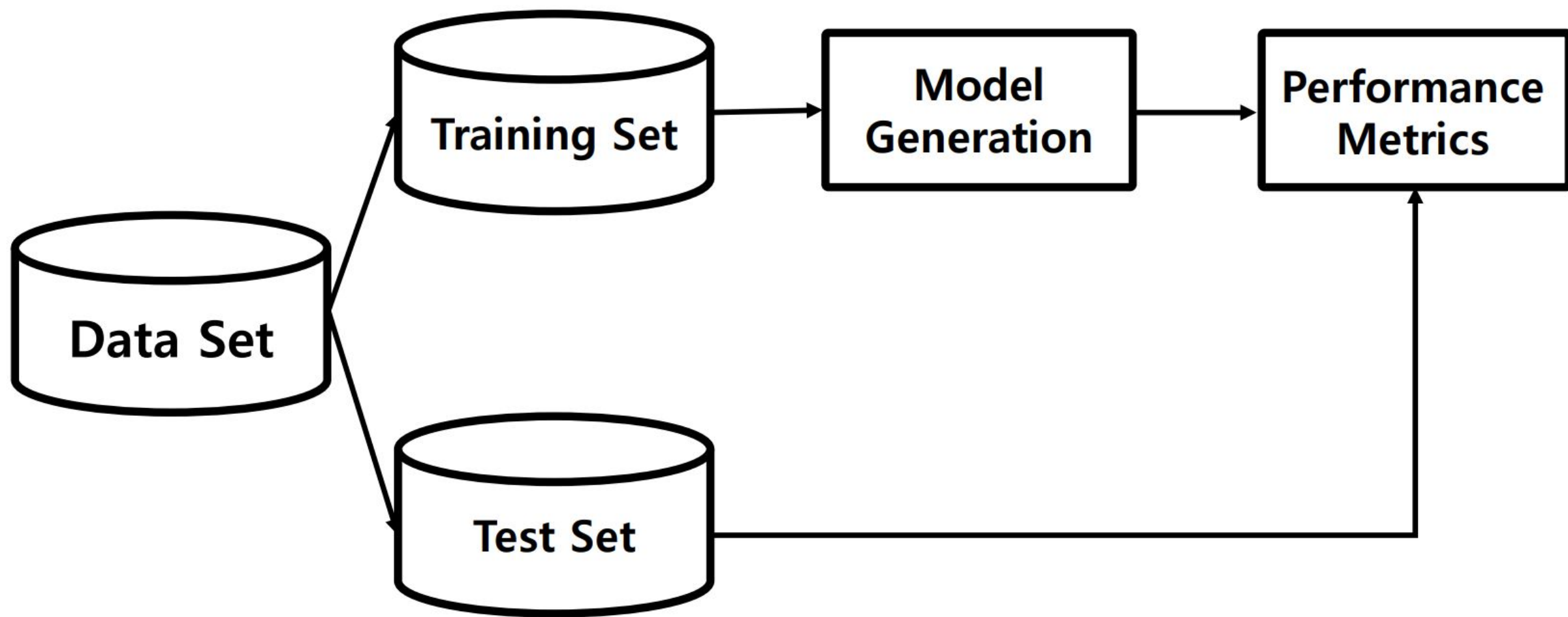


Overcoming **Overfitting**

- 더 많은 데이터를 활용한다.
- Feature의 개수를 줄인다.
- 적절히 Parameter를 선정한다.
- Regularization L1, L2

General ML Process

Training/Test Set



Holdout Method (sampling)

- 데이터를 Training과 Test와 나눠서 모델을 생성하고 테스트하는 기법
- 가장 일반적인 모델 생성을 위한 데이터 랜덤 샘플링 기법
- Training과 Test를 나누는 비율은 데이터의 크기에 따라 다름

Training - Validation - Test

- **Training**

- Model Building

- **Validation**


- Model Check

- **Test**

- Model Evaluation

Validation Set

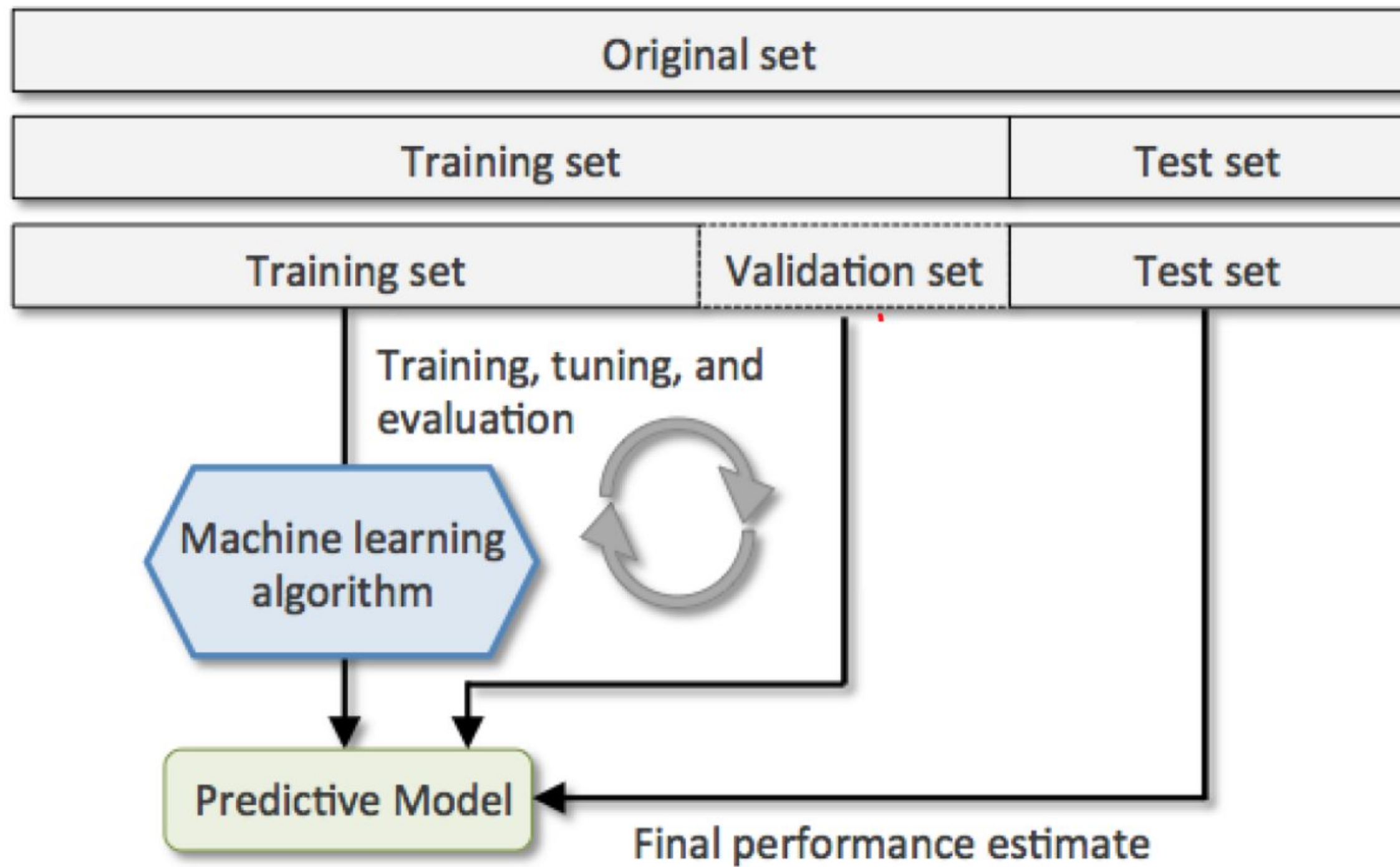
- Test Set은 Model이 생성시 절대 Training Set에 포함되지 않아야 함
- Test Set과 달리 Model 생성시 Model에 성능을 평가하기 위해 사용
- Hyper Parameter Tuning 시 성능 평가를 통해 Overfitting 방지
- Training 중간에 Model의 조정

tune 미국식 [tu:n]  영국식 [tju:n]

1. 곡, 곡조, 선율
2. (악기의) 음을 맞추다, 조율하다
3. (기계를) 조정하다

Training – Validation - Test

6	2	2
Training Set	Validation Set	Test Set



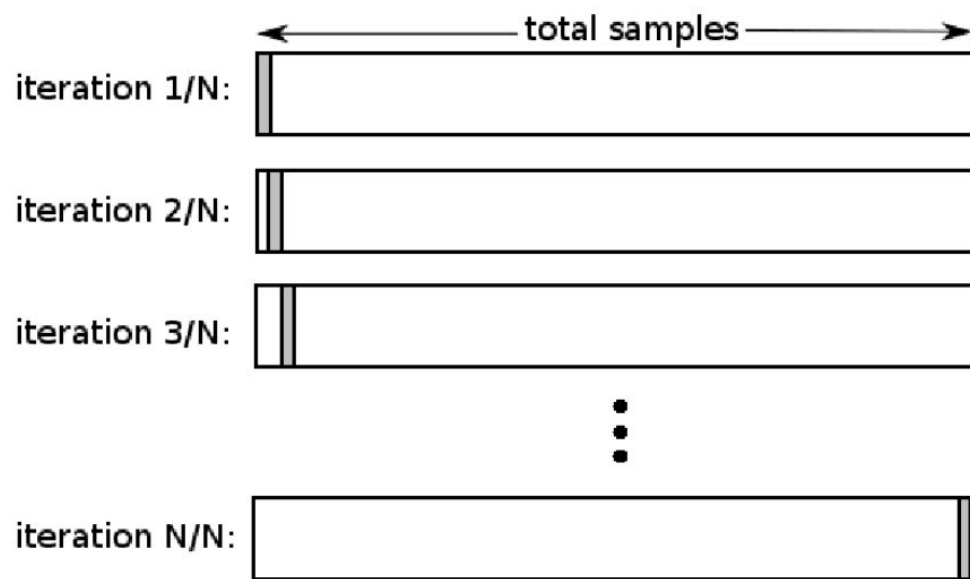
K-fold cross validation

- 학습 데이터를 K번 나눠서 Test와 Train을 실시
 - Test의 평균값을 사용 Training Set Validation Set Validation Set Validation Set Validation Set
- 모델의 Parameter 튜닝, 간단한 모델의 최종 성능 측정 등 사용

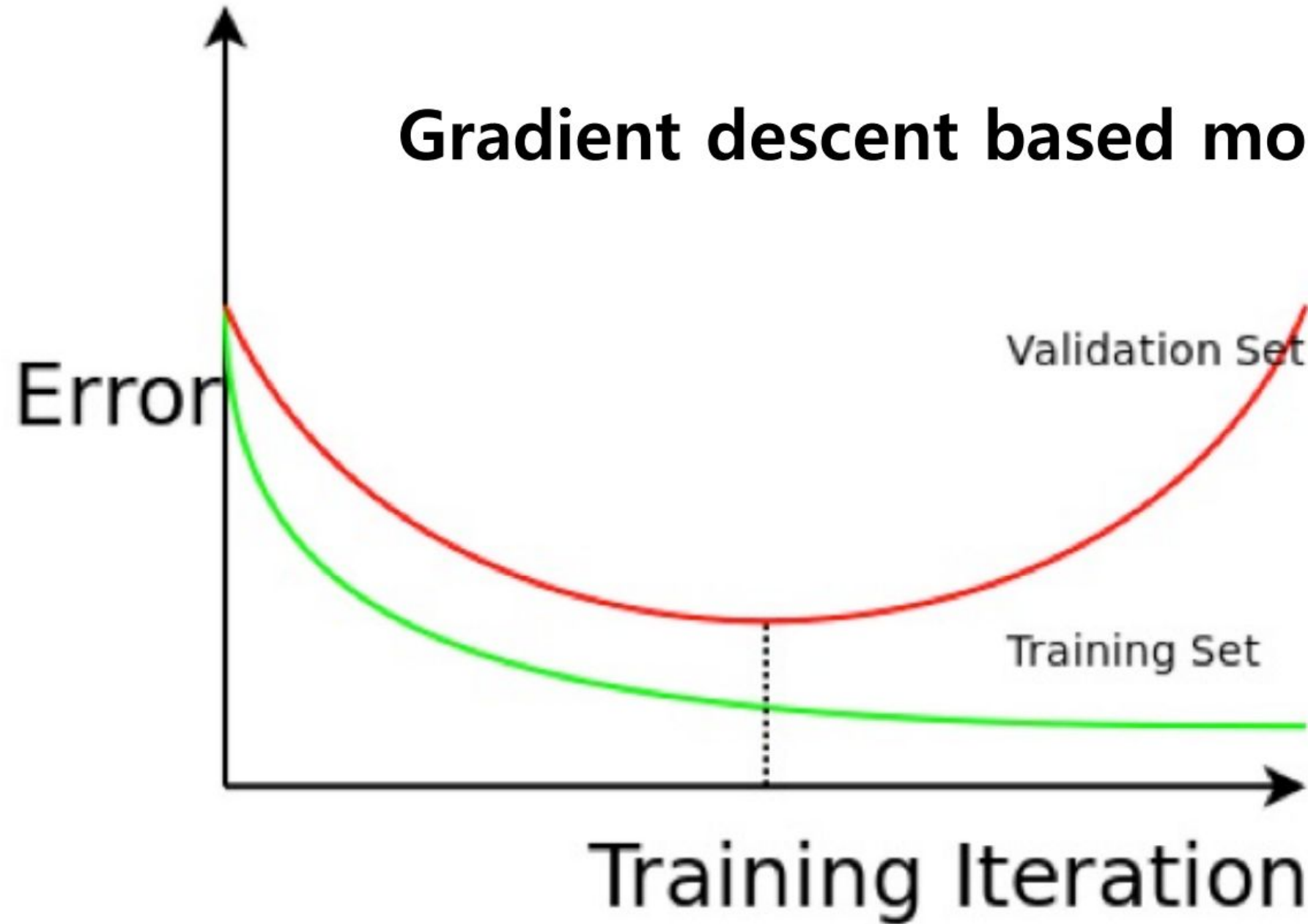
Training Set		Validation Set
	Validation Set	
	Validation Set	
Validation Set		

Leave One Out (LOO)

- Cross validation 에서 test data size가 1인 특별한 경우
- 한번에 한 개의 데이터만 Test set으로 사용함
 - -> 총 k번 iteration



Gradient descent based model



Validation set for parameter Tuning

