

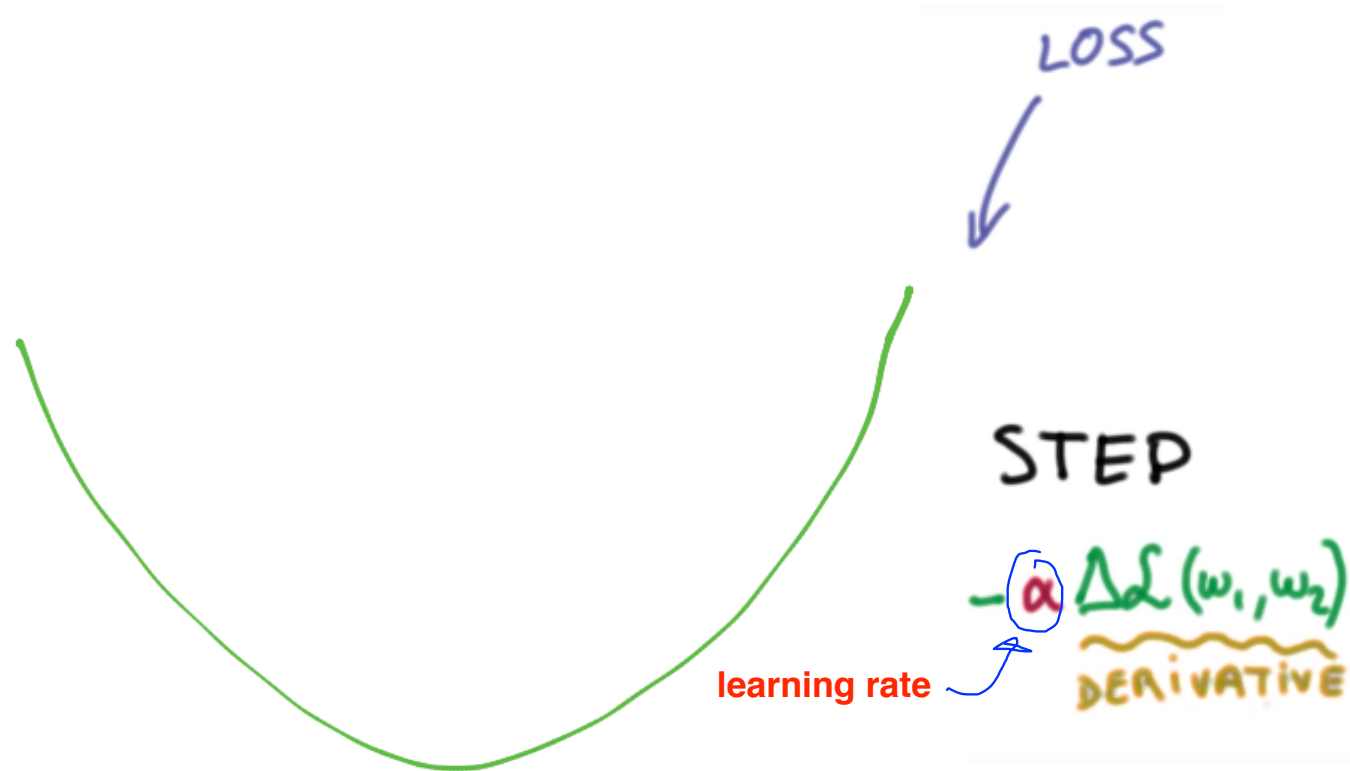
# Lecture 7-I

Application & Tips:

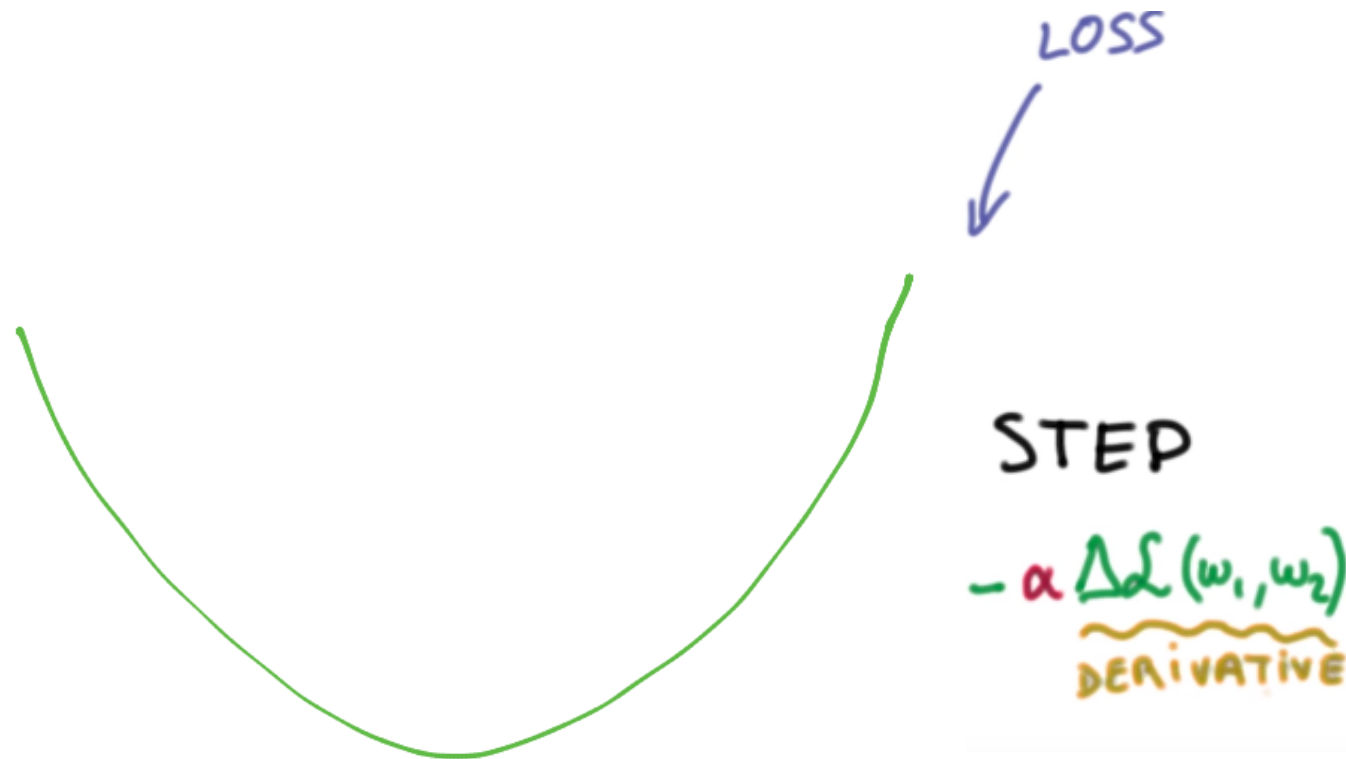
Learning rate, data preprocessing,  overfitting

Sung Kim <hunkim+mr@gmail.com>

# Gradient descent

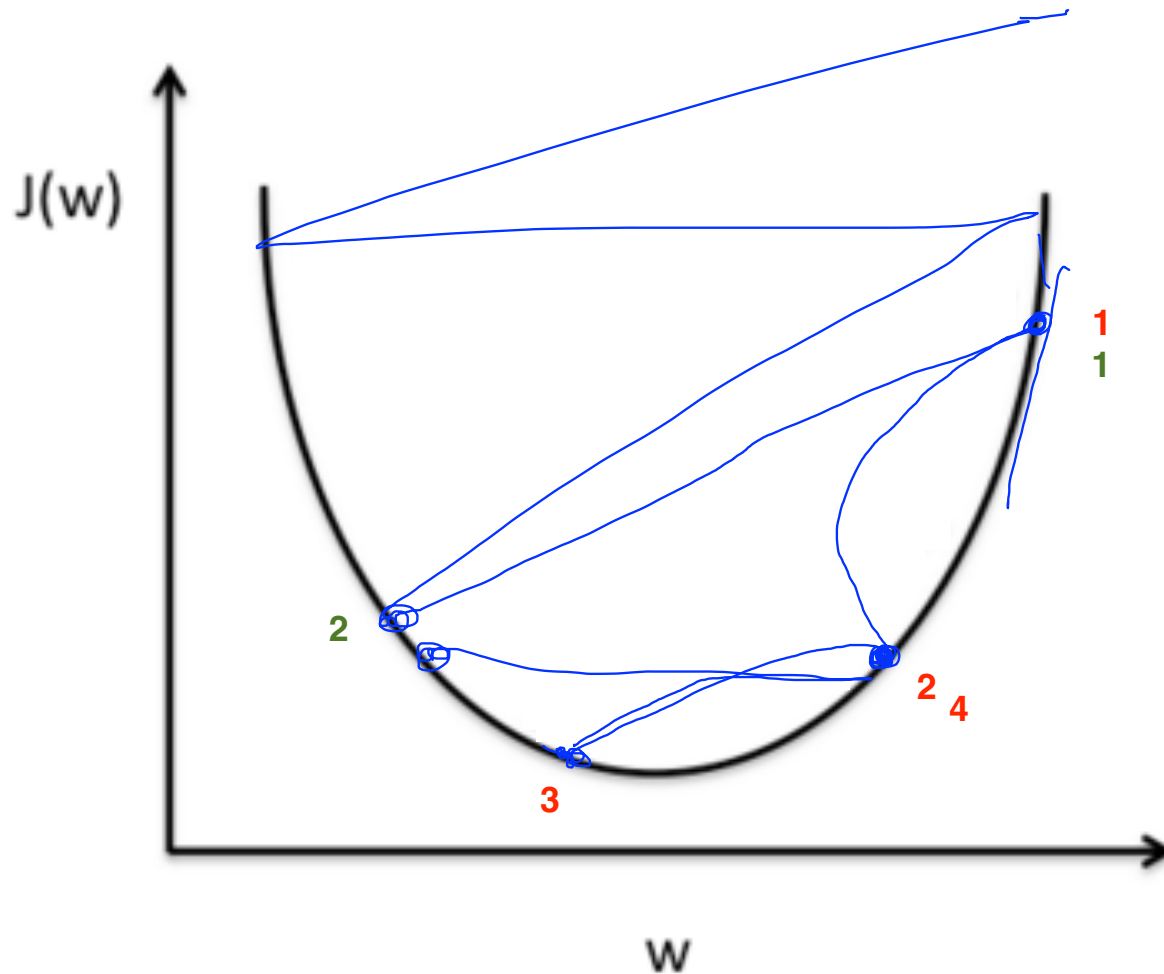


```
# Minimize error using cross entropy
learning_rate = 0.001
cost = tf.reduce_mean(-tf.reduce_sum(Y*tf.log(hypothesis), reduction_indices=1)) # Cross entropy
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost) # Gradient Descent
```



경사면을 내려가는 step

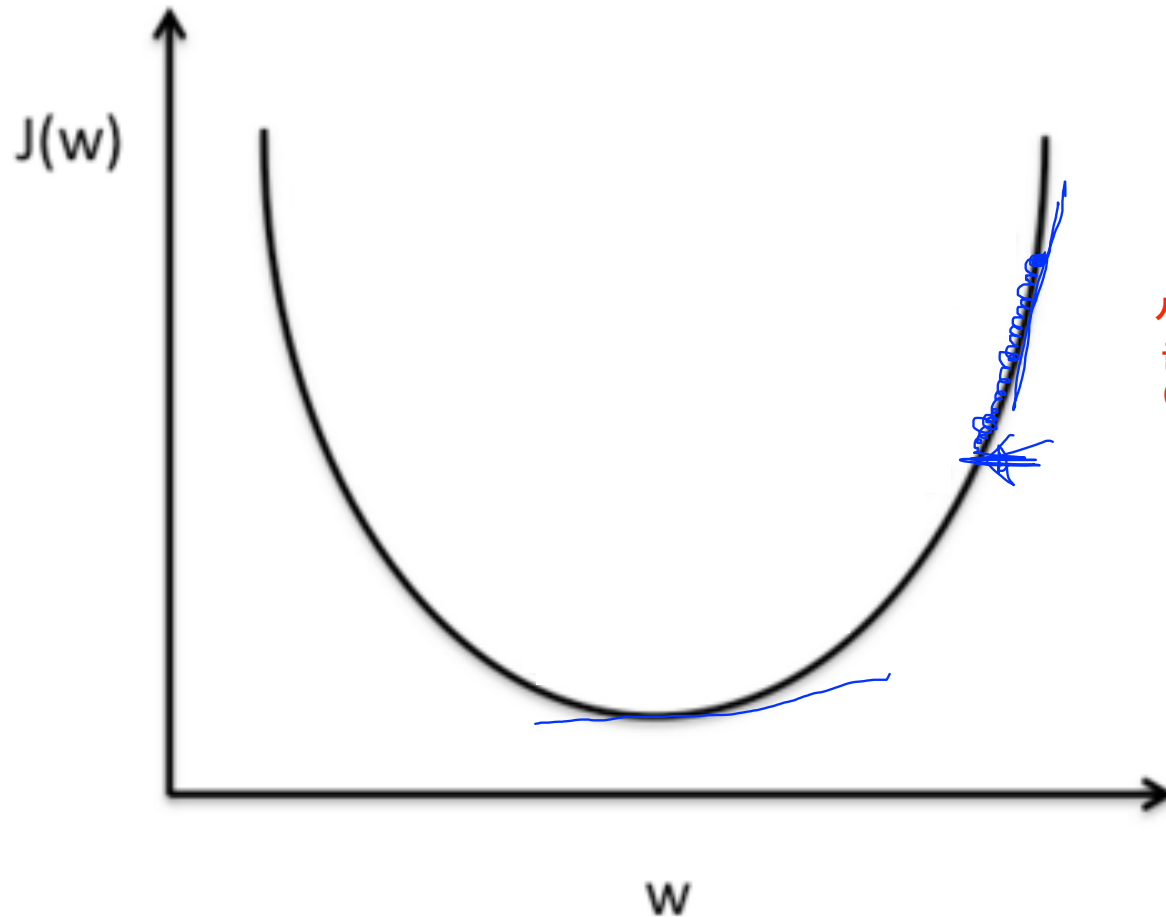
# Large learning rate: overshooting



learning rate가 너무 크면 1번에서 시작했다가 2번으로 갔는데, 최저점을 넘어 3번으로 갔다가 다시 4번으로 가서 global minimum을 못 찾을 수도 있다. 또한 녹색과 같이 1번에서 시작했다가 2번으로 간 후 중간을 왔다 갔다 거리다가 튕겨져 나갈 수도 있다.

굉장히 작은 learning rate 을 준 경우

# Small learning rate: takes too long, stops at local minimum



시간이 오래 걸려서 시간에 제한을 준 경우 global minimum에 도달하지 못했지만 stop 할 수도 있다

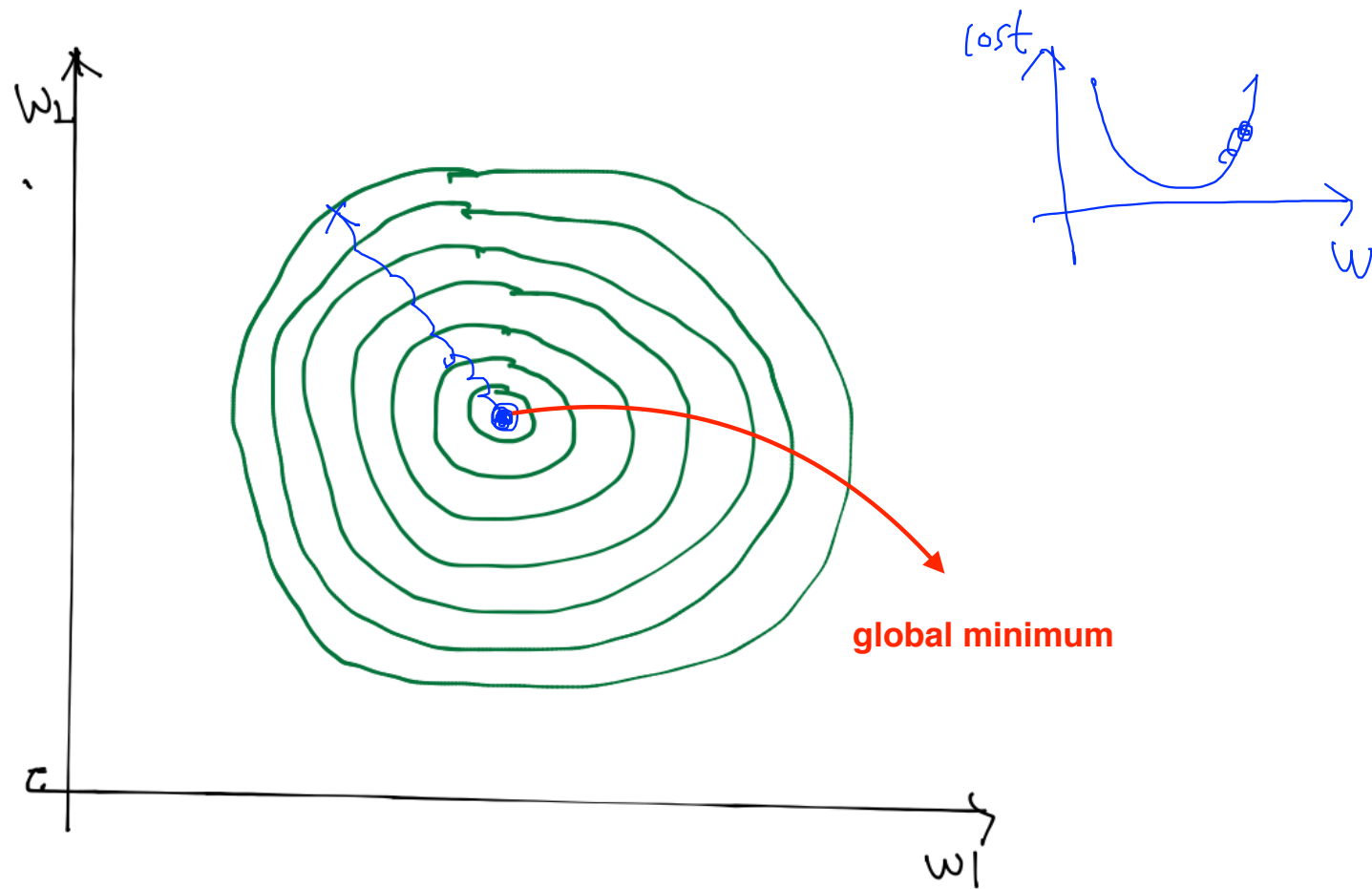
# Try several learning rates

0.01

- Observe the cost function
- Check it goes down in a reasonable rate

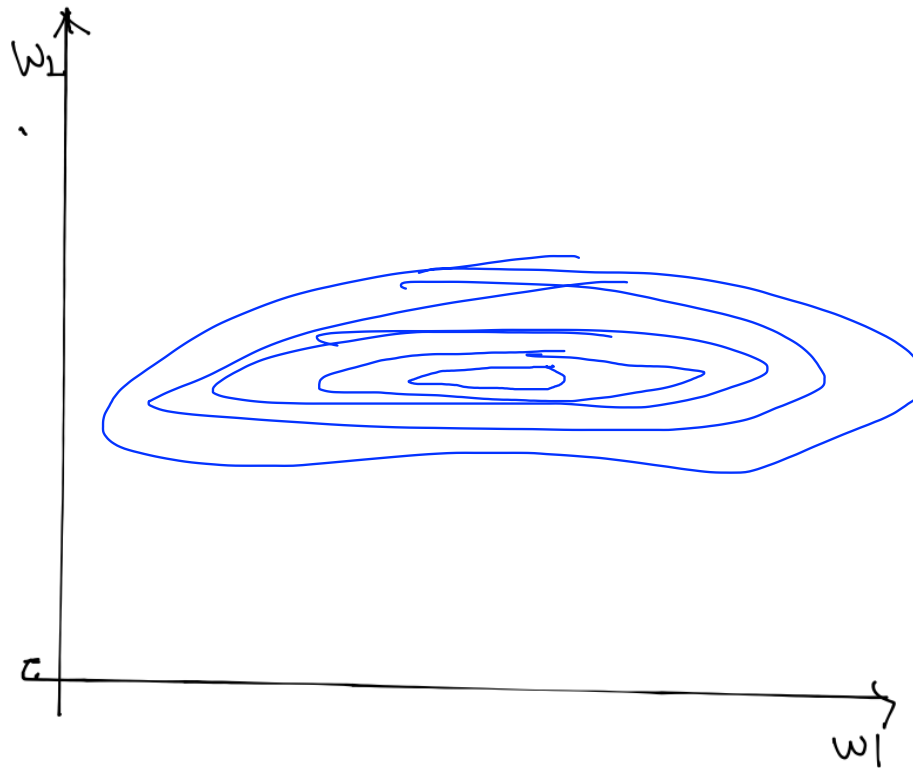
결론, **learning rate** 을 정하는 데 답이 있는 것은 아니다  
작은 값, 큰 값 입력해보면서 확인

# Data (X) preprocessing for gradient descent



# Data (X) preprocessing for gradient descent

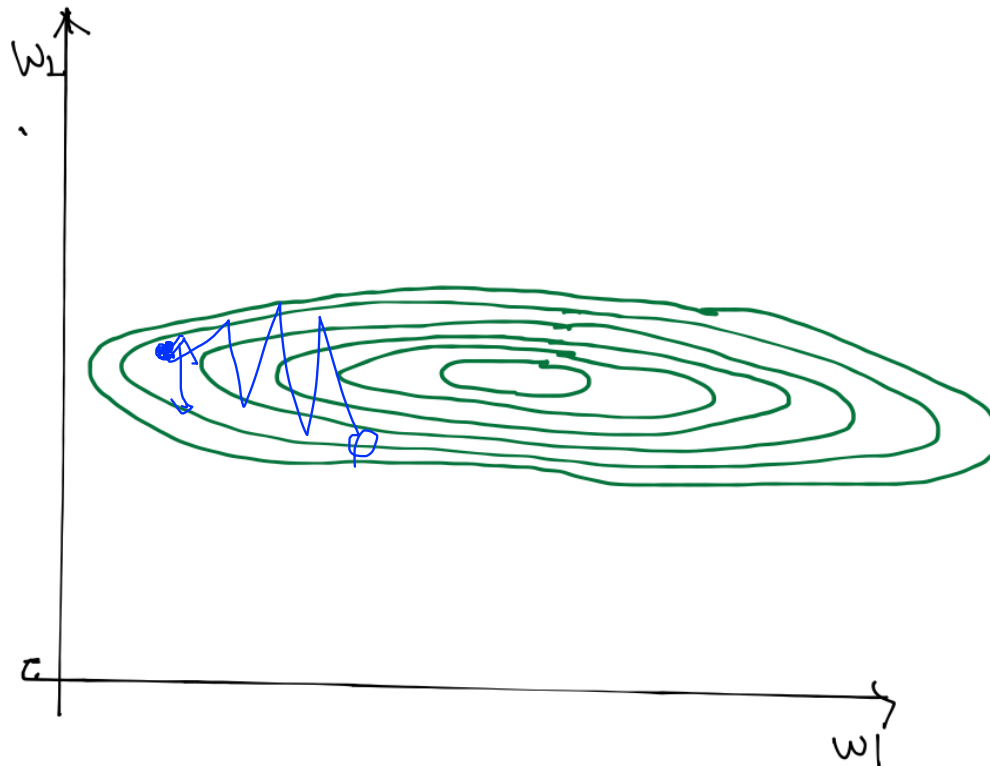
x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C





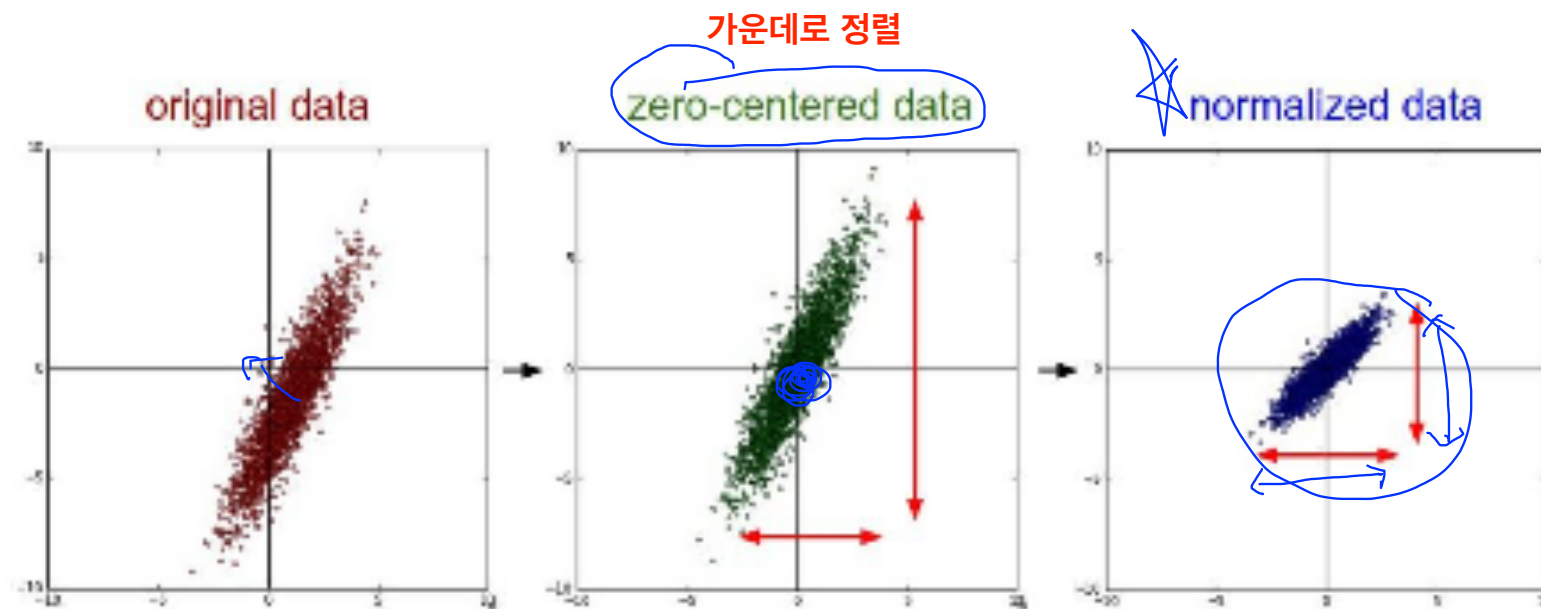
# Data (X) preprocessing for gradient descent

x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C



이렇게 데이터 값에 큰 차이가 있을 경우, **learning rate** 을 잘 설정했음에도 불구하고 튀어나갈 수 있다

# Data (X) preprocessing for gradient descent



# Standardization

$$\boxed{x'_j = \frac{x_j - \mu_j}{\sigma_j}} \quad \star$$

```
x_std[:,0] = (X[:,0] - X[:,0].mean()) / X[:,0].std()
```

[http://sebastianraschka.com/Articles/2015\\_singlelayer\\_neurons.html](http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html)

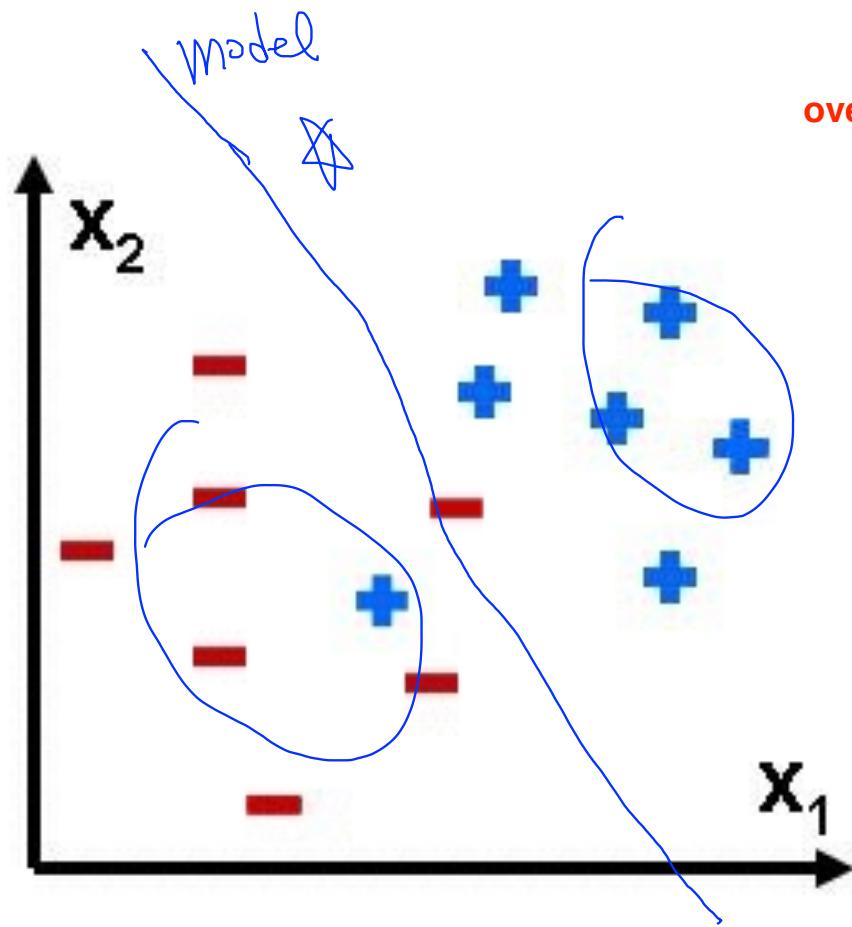
머신 러닝의 가장 큰 문제

# Overfitting

- Our model is very good with training data set (with memorization)  
학습 데이터
- Not good at test dataset or in real use

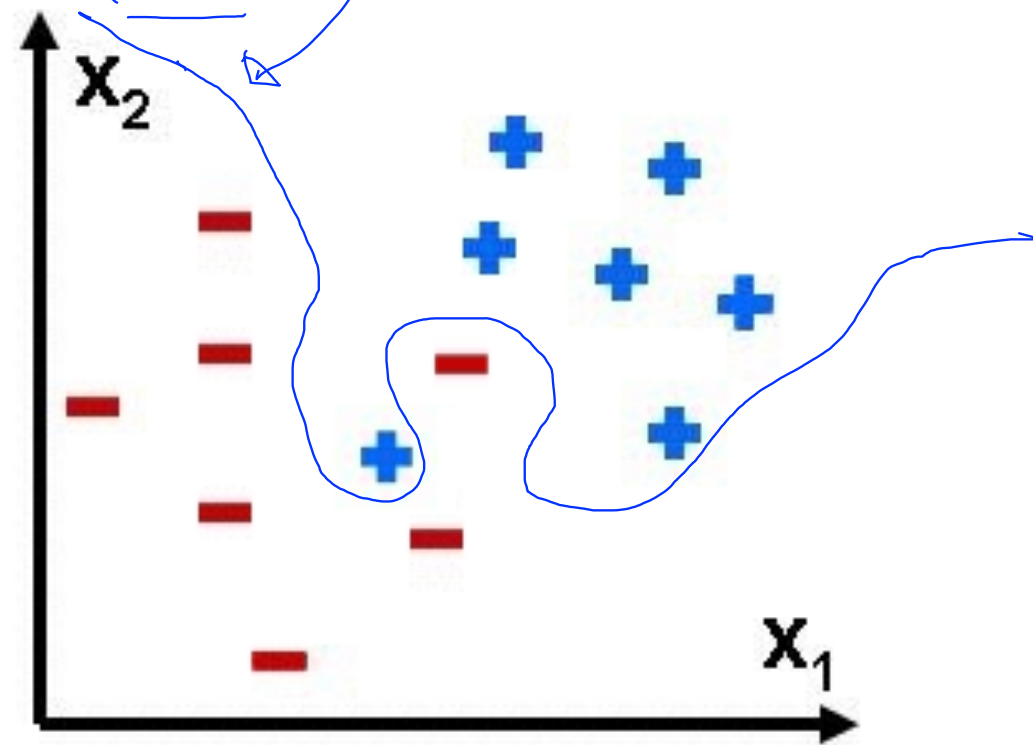
학습 데이터에 너무 잘 맞는 모델이라 실제. 데이터에는 잘 안되는 것

# Overfitting



좋은 모델

overfitting 된 모델 model<sub>2</sub>



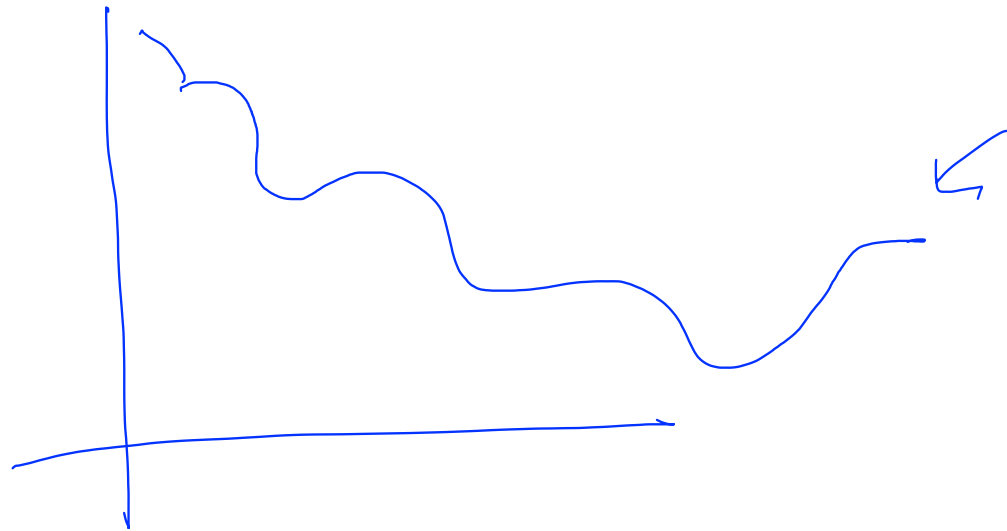
# Solutions for overfitting

- More training data! training 데이터를 많이 가지면 가질수록 overfitting 을 줄일 수 있다
- Reduce the number of features
- Regularization



# Regularization

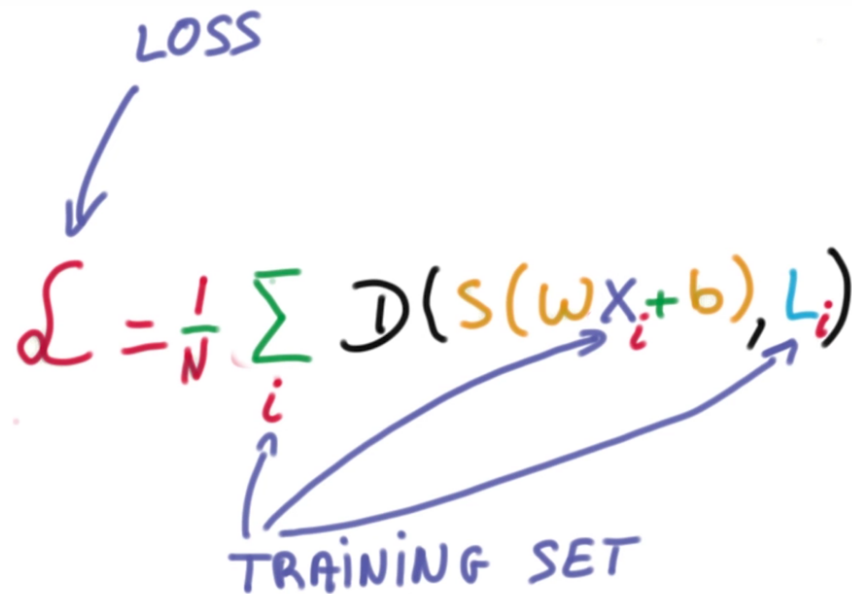
- Let's not have too big numbers in the weight



decision boundary를 데이터에 맞게 구부리는 것을 **overfitting** 이라고 한다  
Weight이 작은 값을 가지면 펴지고, 큰 값을 가지면 구부러진다

# Regularization

- Let's not have too big numbers in the weight



A handwritten diagram illustrating the loss function. The equation is  $\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i)$ . The word "LOSS" is written in blue above the equation, with a blue arrow pointing down to the  $\mathcal{L}$  term. The word "TRAINING SET" is written in blue below the equation, with two blue arrows pointing up to the  $x_i$  and  $L_i$  terms. The terms in the equation are color-coded:  $\mathcal{L}$  is red,  $\frac{1}{N}$  is green,  $\sum$  is green,  $\mathcal{D}$  is black,  $s$  is orange,  $w$  is orange,  $x_i$  is blue,  $+$  is black,  $b$  is orange,  $,$  is black, and  $L_i$  is blue.

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i)$$



# Regularization

- Let's not have too big numbers in the weight

LOSS

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$$

TRAINING SET

regularization strength 상수

이 값이 커지면

0 X

0.001

regularization 을 중요하게 생각한다

# Regularization

- Let's not have too big numbers in the weight

LOSS

`l2reg = 0.001 * tf.reduce_sum(tf.square(W))`

Regularization Strength

$$\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$$

TRAINING SET

The diagram illustrates the components of the L2 regularization loss. At the top, a code snippet `l2reg = 0.001 * tf.reduce_sum(tf.square(W))` is shown with a yellow background. A blue box highlights `l2reg`, and a blue arrow points from it to the  $\lambda$  term in the formula below. Another blue arrow points from the `0.001` value to the  $\lambda$  term. A third blue arrow points from the `tf.reduce_sum(tf.square(W))` part to the  $\sum W^2$  term. The formula  $\mathcal{L} = \frac{1}{N} \sum_i \mathcal{D}(s(w x_i + b), L_i) + \lambda \sum W^2$  is written in red and green. A blue arrow points from the word 'TRAINING SET' at the bottom to the  $\sum_i$  part of the formula. A blue arrow also points from the word 'LOSS' at the top left to the  $\mathcal{L}$  term. The word 'Regularization Strength' is written in blue above the  $\lambda$  term.

# Summary

- Learning rate ✓
- Data preprocessing
- Overfitting ✗
  - More training data
  - Regularization