# Regression

# Linear regression

- Linear regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Predicted value

Bias(편향)

Feature value

Modell parameter
(weight)

- Vectorized form

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

Feature vecture

Parameter vector

Hypothesis function
(using the model parameter theta)

- Training the model = Setting the model parameter

# MSE cost function

- Find the theta to minimize the MSE

$$\mathrm{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(i)} - y^{(i)}\right)^2 \longrightarrow$$

(Predicted value - real value)

# Normal equation

$$\widehat{\boldsymbol{\theta}} = \left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1} \mathbf{X}^{\top} \mathbf{y}$$

$\downarrow$

Minimized theta

# Normal equation 증명 (추가 설명)

$(A^T)^T = A$

$(A+B)^T = A^T + B^T$

$(AB)^T = B^T A^T$

$(kA)^T = kA^T$ ($k$는 임의의 상수)

$A^T B = B^T A$

전치 행렬의 기본 성질

$$y = Wx$$

$$\boxed{cost} = \frac{1}{m}\sum_{i=1}^{m}(\boxed{H(x^{(i)})} - y^{(i)})^2$$

$$\boxed{MSE(W)} = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

전치 행렬 x 행렬 = 제곱의 합

① $= \frac{1}{m}((WX - y)^T(WX - y))$

② $= \frac{1}{m}(((WX)^T - y^T)(WX - y))$

③ $= \frac{1}{m}((WX)^T WX - (WX)^T y - y^T WX + y^T y)$

④ $= \frac{1}{m}(X^T W^T WX - 2(WX)^T y + y^T y)$

① $MSE(W) = \frac{1}{m}(X^T \boxed{W^T W} X - 2\boxed{(WX)^T} y + y^T y)$

② $MSE(W) = \frac{1}{m}(X^T X \boxed{W^2} - 2\boxed{X^T} y \boxed{W^T} + y^T y)$

① $MSE(W) = \frac{1}{m}(X^T X W^2 - 2X^T y W^T + y^T y)$

② $\frac{dMSE(W)}{dW} = \frac{1}{m}(2X^T XW - 2X^T y) = 0$

③ $\frac{dMSE(W)}{dW} = 2X^T XW - 2X^T y = 0$

$$2X^T XW - 2X^T y = 0$$

$$2X^T XW = 2X^T y$$

$$X^T XW = X^T y$$

$$W = (X^T X)^{-1} X^T y$$

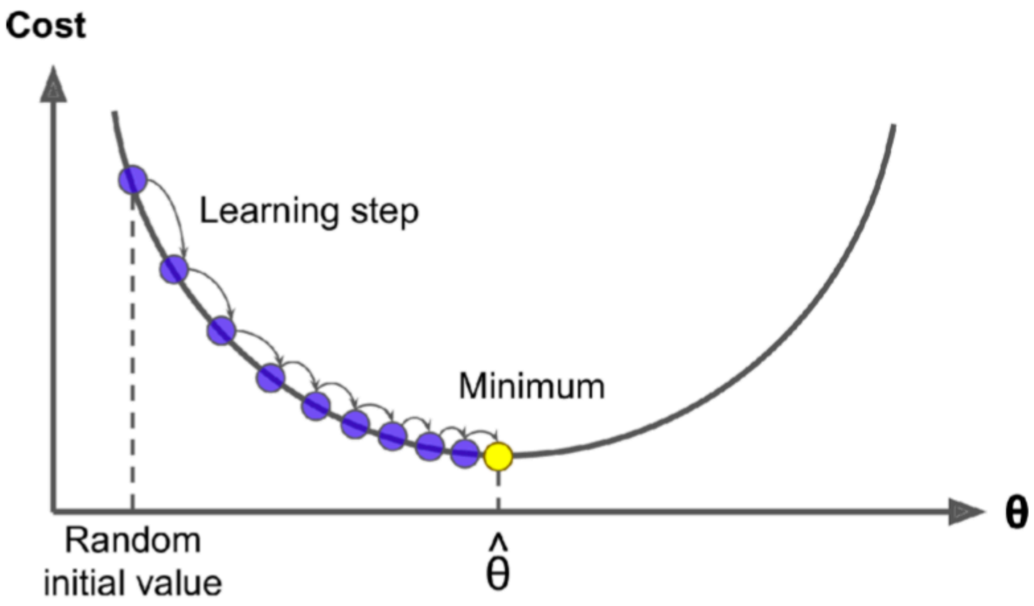
# Pseudoinverse (유사역행렬)

$$X^+ = V\Sigma^+U^\top$$

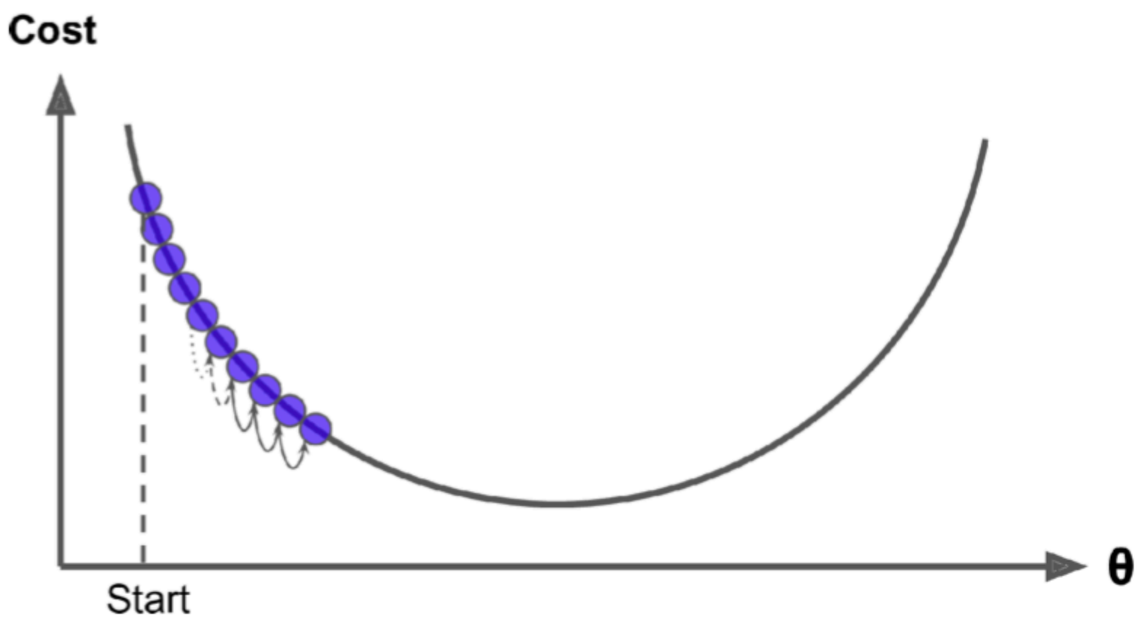
- Use SVD (특이값 분해)
- the algorithm takes **Σ**
- sets to zero all values smaller than a tiny threshold value
- this approach is more efficient than computing the normal equation

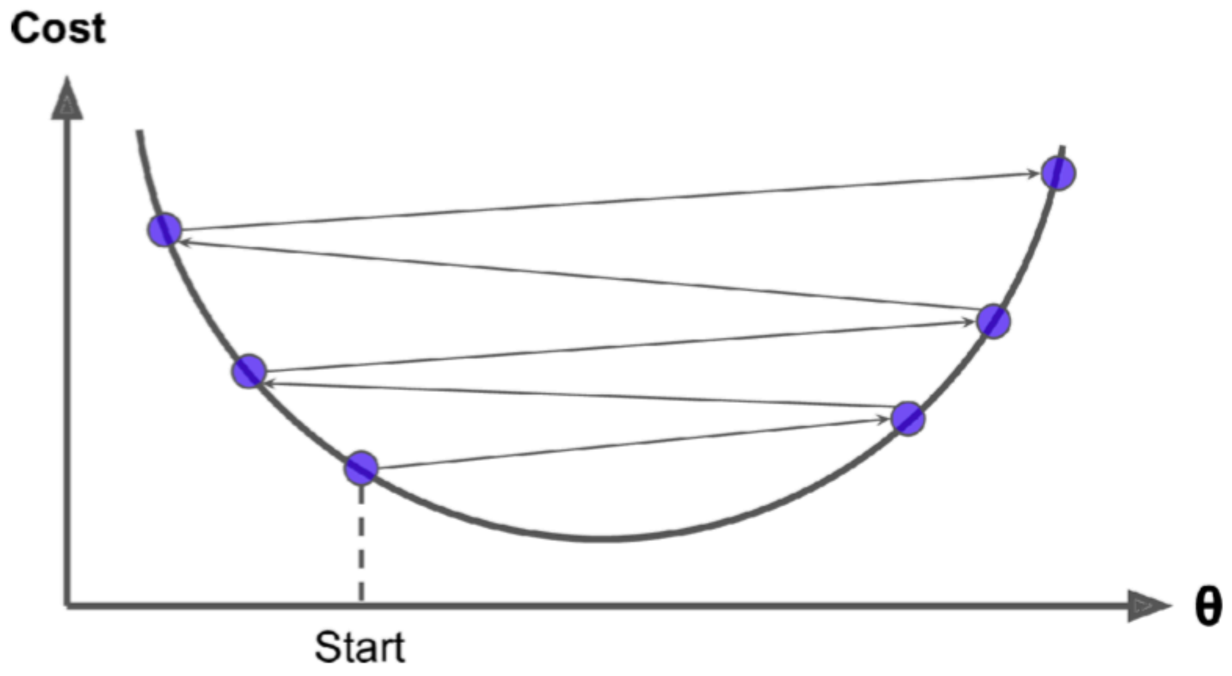
# Gradient descent

- Tweak parameters iteratively in order to minimize a cost function
- Random initialization (무작위 초기화)
- Determined by the learning rate hyperparameter



# Gradient descent



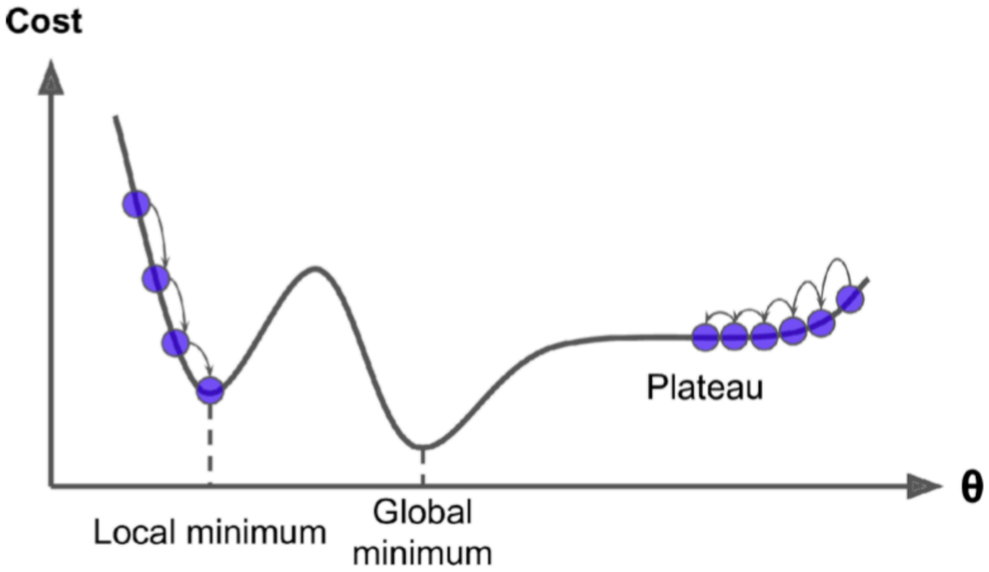




- If learning rate is too small,Need many time

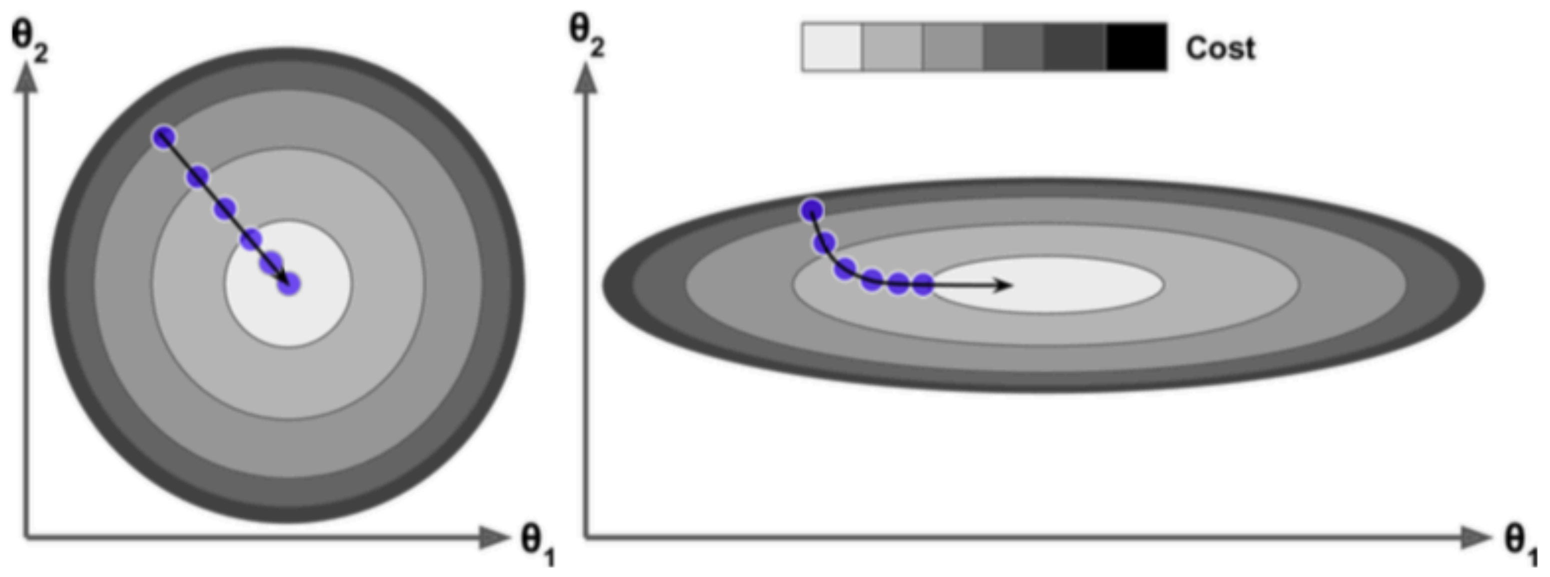
- If learning rate is too big, Make the algorithm diverge, with larger and larger values

# Gradient descent

- Not all cost functions look like nice, regular bowls
- It will converge to a local minimum, which is not as good as the global minimum
- The mse cost function for a. inear regression model happens to be a convex function



# Gradient descent




Feature scaling => fast

Without feature sacling => slow

- Training a model means searching for a combination of model parameters that minimize a cost function
- It is a search in the model's parameter space

# Batch Gradient descent (배치 경사 하강법)

- Partial derivative(편도 함수):

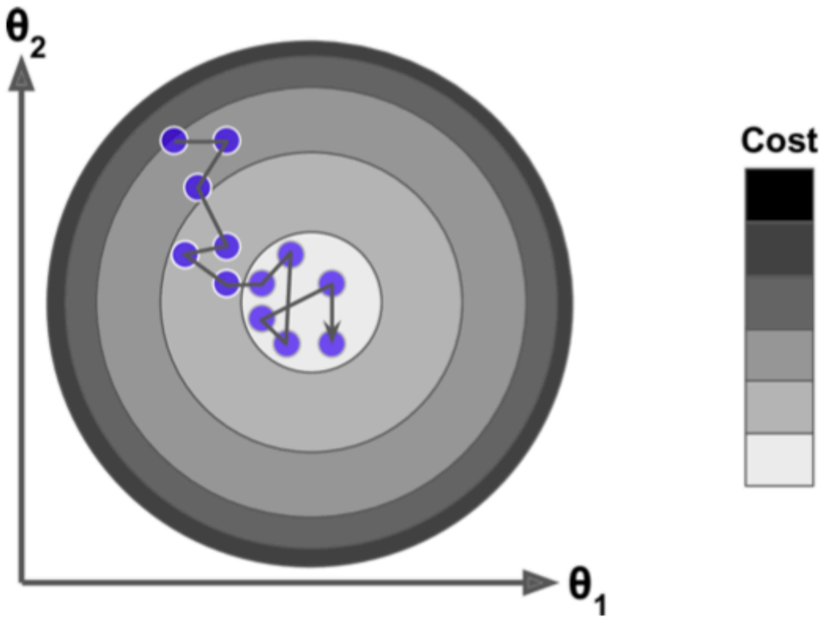
$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\boldsymbol{\theta}) = \frac{2}{m} \sum_{i=1}^{m} \left( \boldsymbol{\theta}^\top \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

- Step of gradient descent

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

# SGD (확률적 경사 하강법)

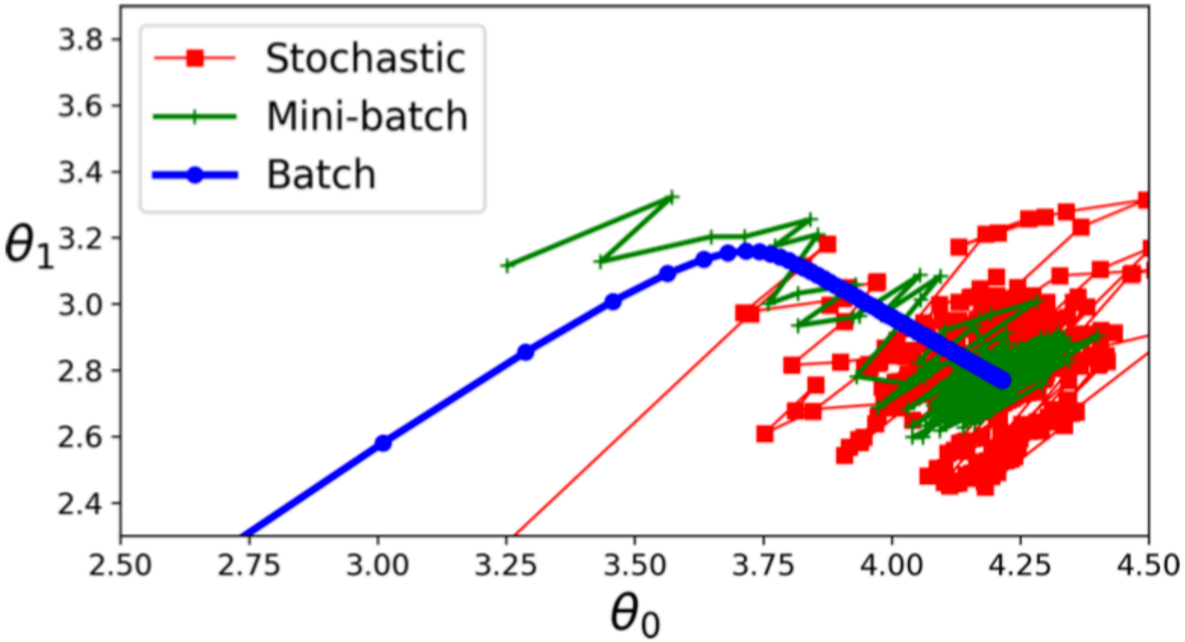



- Faster than batch gradient descent
- Less regular than Batch Gradient Descent
- Jump out of a local minima, but bad because it means that the algorithm can never settle at the minimum

-> solution: reduce the learning rate

# Mini-batch gradient descent (미니 배치 경사 하강법)

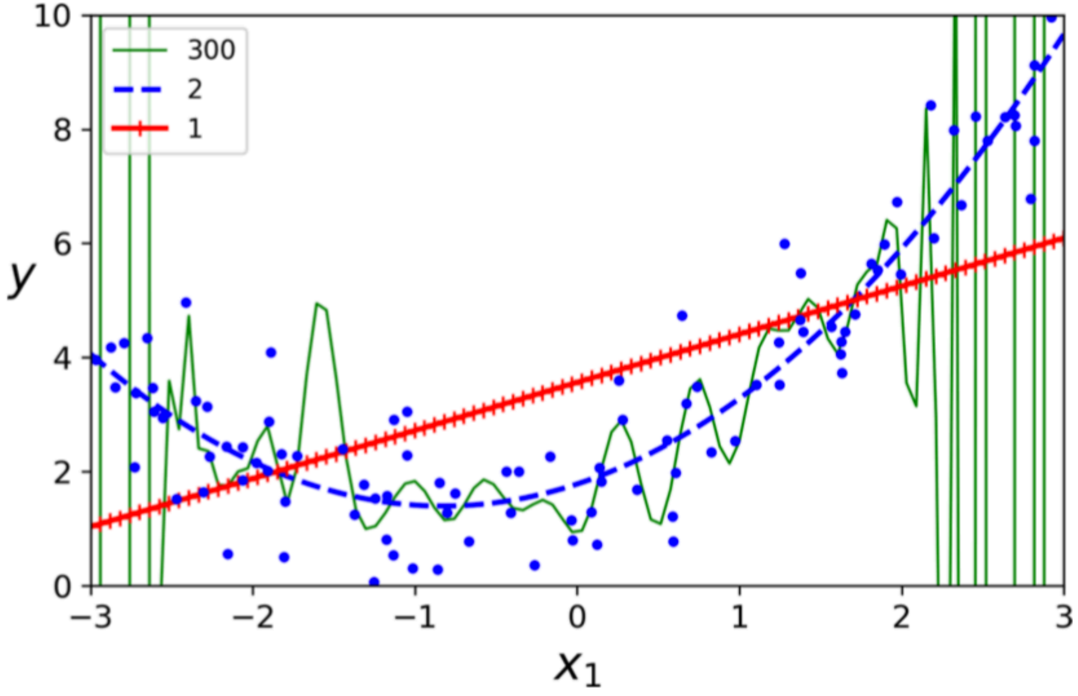
- Mimi-batch (small sample set instead of one sample)



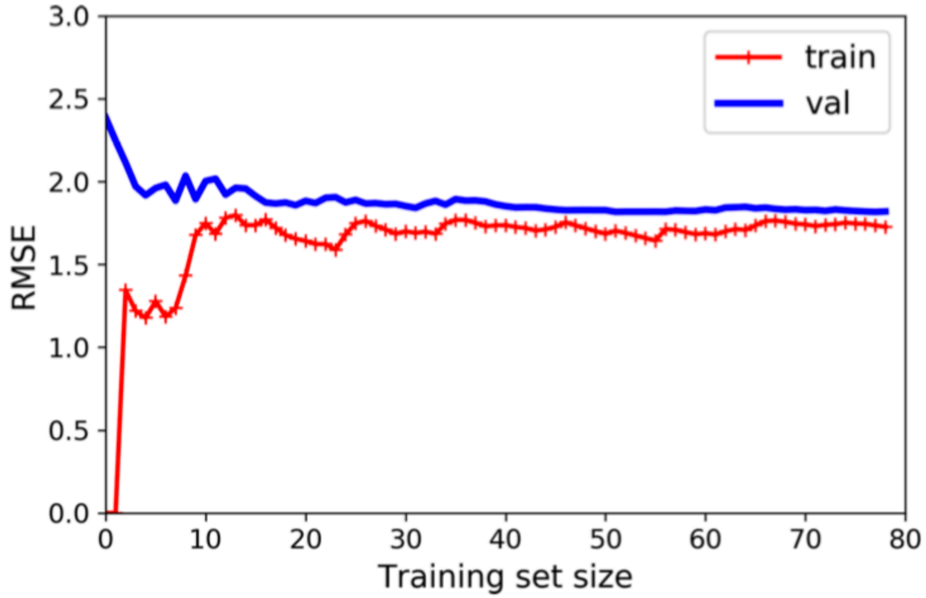
# Polynomial regression (다항 회귀)

- High-degree Polynomial Regression model is severly overfitting data

-> The model should be generalized



# Polynomial regression (다항 회귀)




- small training set size

-> The model can fit them perfectly

- large training set size

-> The error on the training data goes up

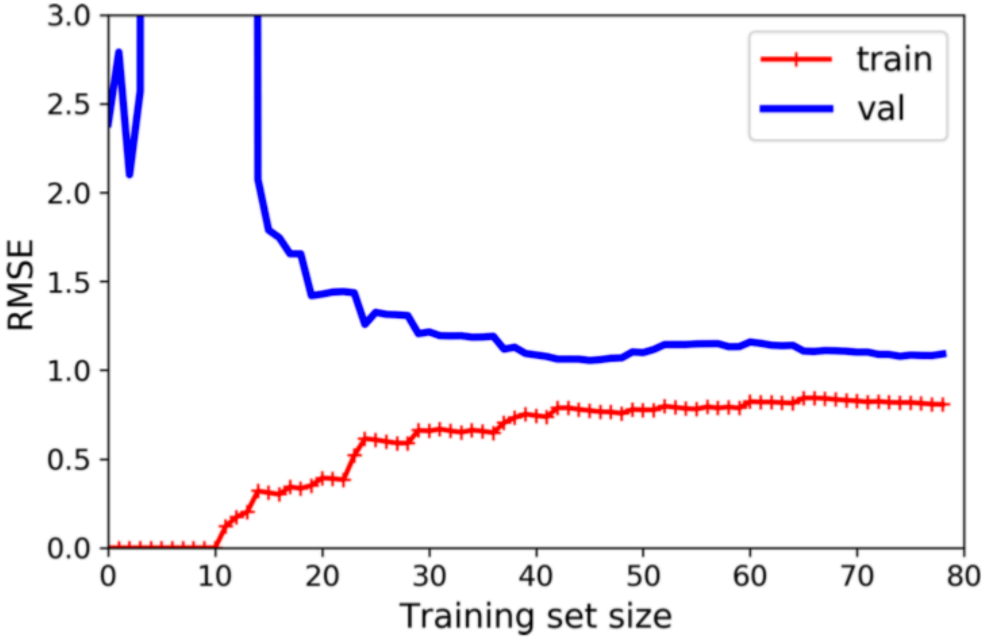
- small validation set size

-> Incapable of generalizing properly

- Large validation set size

-> Validation error slowly goes down

# Polynomial regression (다항 회귀)




- the error on the training data is  lower than with the Linear Regression model

- There is a gap between the curves

-> This means overfitting

## Regularized Linear Models

- A good way to reduce overfitting is to regularize the model

## Ridge Regression (릿지 회귀)

- Ridge Regression cost function

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha\frac{1}{2}\Sigma_{i=1}^{n}\theta_i^2$$

- Normal equation

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{X}^\top\mathbf{X}\boxed{+\alpha\mathbf{A}}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

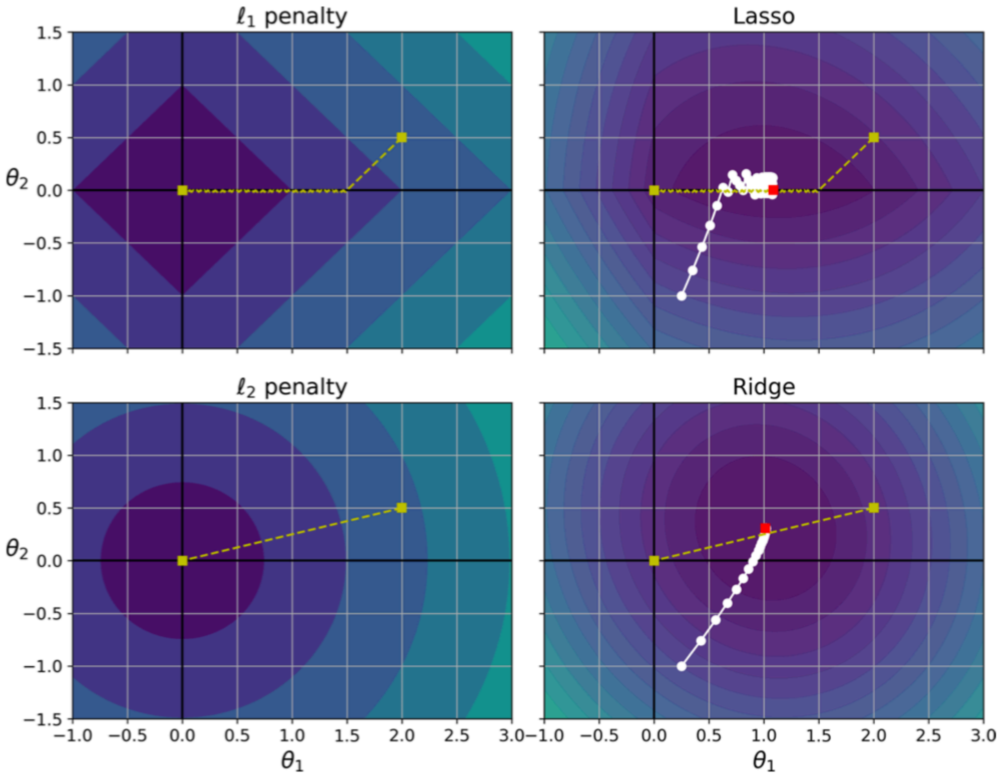
# Lasso Regression (라쏘 회귀)

- Lasso Regression cost function

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \Sigma_{i=1}^{n} \boxed{|\theta_i|}$$

- subgradient vector

$$g(\boldsymbol{\theta}, J) = \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \begin{pmatrix} \mathrm{sign}\,(\theta_1) \\ \mathrm{sign}\,(\theta_2) \\ \vdots \\ \mathrm{sign}\,(\theta_n) \end{pmatrix} \quad \text{where } \mathrm{sign}\,(\theta_i) = \begin{cases} -1 & \text{if } \theta_i < 0 \\ 0 & \text{if } \theta_i = 0 \\ +1 & \text{if } \theta_i > 0 \end{cases}$$

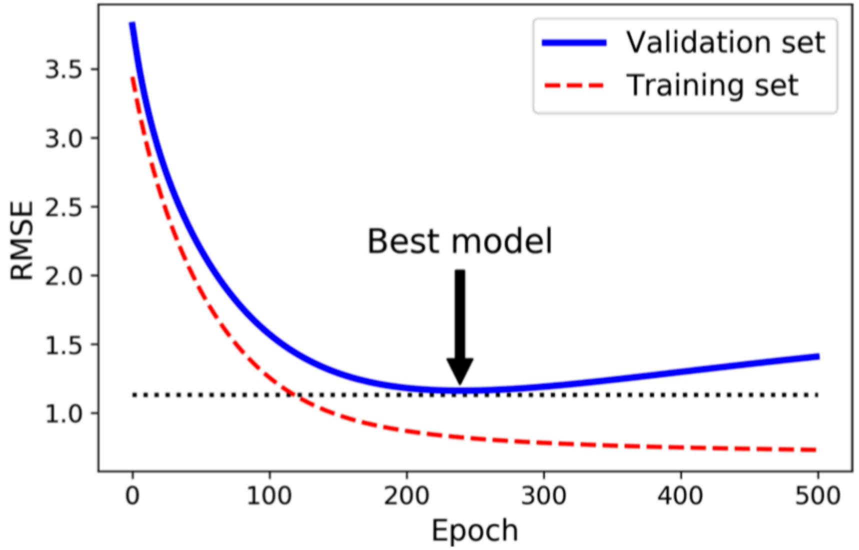


# Elastic net (엘라스틱 넷)

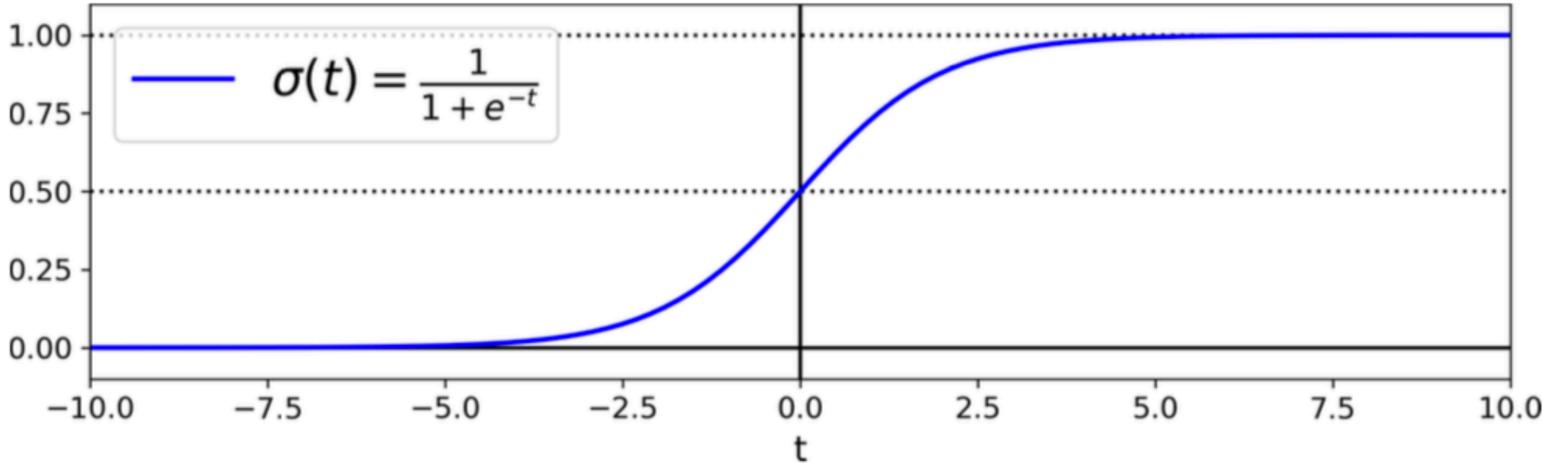
$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + r\alpha\Sigma_{i=1}^{n}\left|\theta_i\right| + \frac{1-r}{2}\alpha\Sigma_{i=1}^{n}\theta_i^{2}$$

# Early stopping (조기 종료)

- Regularize iterative learning algorithms such as Gradient Descent is to stop training as soon as the validation errors reaches a minimum



# Logistic Regression




$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

# Logistic Regression

- cost function of logistic regression

t

$$c(\boldsymbol{\theta}) = \begin{cases} -\log\left(\boxed{\hat{p}}\right) & \text{if } y = 1 \\ -\log\left(1 - \hat{p}\right) & \text{if } y = 0 \end{cases}$$

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\Sigma_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)log\left(1 - \hat{p}^{(i)}\right)\right]$$

- logistic cost function partial derivatives(편도 함수)

$$\frac{\partial}{\partial \theta_j}\mathrm{J}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^{\top}\mathbf{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

# Softmax Regression (multinomial logistic regression)

- softmax function

$$\hat{p}_k = \sigma(\mathbf{s}(\mathbf{x}))_k = \frac{\exp\left(s_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(s_j(\mathbf{x})\right)}$$

- prediction

$$\hat{y} = \underset{k}{\operatorname{argmax}}\ \sigma(\mathbf{s}(\mathbf{x}))_k = \underset{k}{\operatorname{argmax}}\ s_k(\mathbf{x}) = \underset{k}{\operatorname{argmax}}\ \left(\left(\boldsymbol{\theta}^{(k)}\right)^{\mathsf{T}} \mathbf{x}\right)$$

# Cross entropy

- Cross entropy cost function

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{p}_k^{(i)}\right)$$

- Cross entropy gradient vector for class k

$$\nabla_{\boldsymbol{\theta}^{(k)}} J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{p}_k^{(i)} - y_k^{(i)}\right)\mathbf{x}^{(i)}$$