



Machine Learning 09

Kihyun Shin
DMSE, HBNU

Back propagation

Derivative example

Cost function

$$J(w) = w^2$$

Let's say $w = 3$

$$J(w) = 3^2 = 9$$

If we increase w by a tiny amount $\varepsilon = 0.001$, how does $J(w)$ change ?

$$w = 3 + 0.001$$

$$J(w) = w^2 = 9.006001$$

If $w \uparrow 0.001$

If $J(w) \uparrow 6 \times 0.001$

ε

$6 \times \varepsilon$

$$\frac{\partial}{\partial w} J(w) = 6$$

$\varepsilon = 0.002$

$$w = 3 + 0.002$$

$$J(w) = w^2 = 9.012004$$

If $w \uparrow 0.002$

If $J(w) \uparrow 6 \times 0.002$

ε

$6 \times \varepsilon$

$$\frac{\partial}{\partial w} J(w) = 6$$

If epsilon is small enough, there will be no problem



Informal Definition of Derivative

If $w \uparrow \varepsilon$ causes $J(w) \uparrow k \times \varepsilon$ then

$$\frac{\partial}{\partial w} J(w) = k$$

Gradient descent

repeat {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

}

If derivative $\rightarrow$ small, then update step $\rightarrow$ small

If derivative $\rightarrow$ large, then update step $\rightarrow$ large

More Derivative Examples

$$w = 3 \quad J(w) = w^2 = 9 \quad w \uparrow 0.001 \quad J(w) = J(3.001) = 9.006001 \quad \frac{\partial}{\partial w} J(w) = 6$$

$$J(w) \uparrow 6 \times 0.001$$

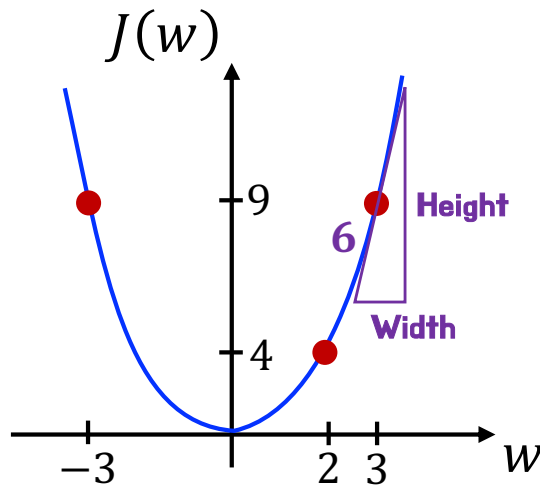
$$w = 2 \quad J(w) = w^2 = 4 \quad w \uparrow 0.001 \quad J(w) = J(2.001) = 4.004001 \quad \frac{\partial}{\partial w} J(w) = 4$$

$$J(w) \uparrow 4 \times 0.001$$

$$w = -3 \quad J(w) = w^2 = 9 \quad w \uparrow 0.001 \quad J(w) = J(-2.999) = 8.994001 \quad \frac{\partial}{\partial w} J(w) = -6$$

$$J(w) \downarrow 6 \times 0.001$$

$$J(w) \uparrow -6 \times 0.001$$



Calculus	w	$\frac{\partial}{\partial w} J(w)$
$\frac{\partial}{\partial w} J(w) = 2w$	3	$2 \times 3 = 6$
	2	$2 \times 2 = 4$
	-3	$2 \times -3 = -6$

Even More Derivative Examples

$w = 2$	$J(w) = w^2 = 4$	$\frac{\partial}{\partial w} J(w) = 2w = 4$	$w \uparrow 0.001$	$J(w) = 4.004001$ $J(w) \uparrow 4 \times 0.001$
	$J(w) = w^3 = 8$	$\frac{\partial}{\partial w} J(w) = 3w^2 = 12$	$w \uparrow \varepsilon$	$J(w) = 8.012006$ $J(w) \uparrow 12 \times \varepsilon$
	$J(w) = w = 2$	$\frac{\partial}{\partial w} J(w) = 1$	$w \uparrow \varepsilon$	$J(w) = 2.001$ $J(w) \uparrow 1 \times \varepsilon$
	$J(w) = \frac{1}{w} = \frac{1}{2}$	$\frac{\partial}{\partial w} J(w) = -\frac{1}{w^2} = -\frac{1}{4}$	$w \uparrow \varepsilon$ $w = \frac{1}{2.001}$	$J(w) = 0.49975$ $= 0.5 - 0.25 \times 0.001$ $J(w) \uparrow -\frac{1}{4} \times \varepsilon$

$$\frac{\partial}{\partial w} J(w) \quad w \uparrow \varepsilon \quad J(w) \uparrow k \times \varepsilon$$



A Note on Derivative Notation

If $J(w)$ is a function of one variable (w),

$$\frac{d}{dw}J(w)$$

If $J(w_1, w_2, \dots, w_n)$ is a function of more than one variable,

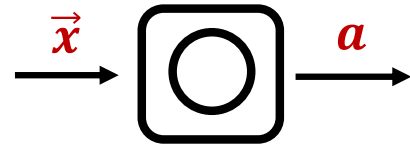
$$\frac{\partial}{\partial w_i}J(w_1, w_2, \dots, w_n) \quad \frac{\partial J}{\partial w_i} \quad \text{or} \quad \frac{\partial}{\partial w_i}J$$

Called “partial derivative”

Notation ∂ will be used in these courses



Small Neural Network Example



$$w = 2$$

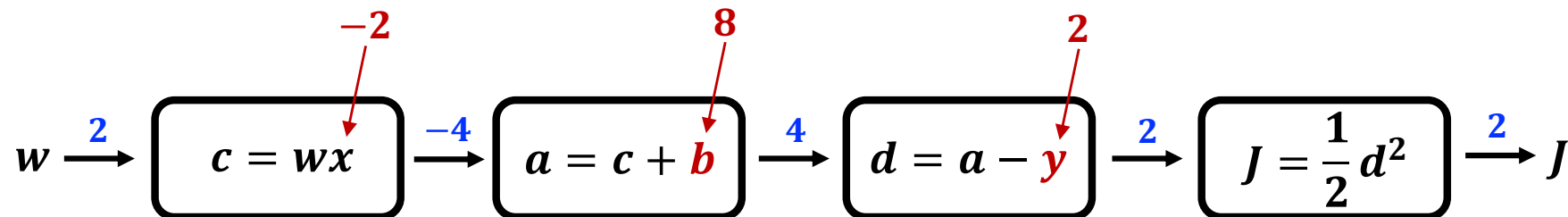
$$x = -2$$

$$b = 8$$

$$y = 2$$

$$a = wx + b \quad : \text{Linear activation } a = g(z) = z$$

$$J(w, b) = \frac{1}{2} (a - y)^2$$



Computation graph

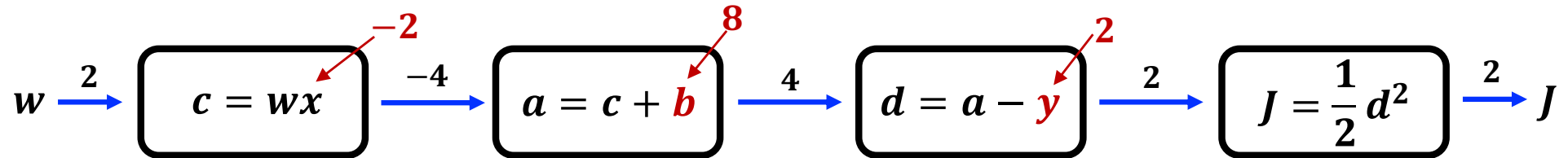
Forward propagation

How do we find the derivatives of J ? $\Rightarrow \frac{\partial J}{\partial w}$ or $\frac{\partial J}{\partial b}$

Computing the Derivatives

 **Forward prop**
 **Back prop**

$$w = 2 \quad b = 8 \quad x = -2 \quad y = 2 \quad a = wx + b \quad J(w, b) = \frac{1}{2} (a - y)^2$$



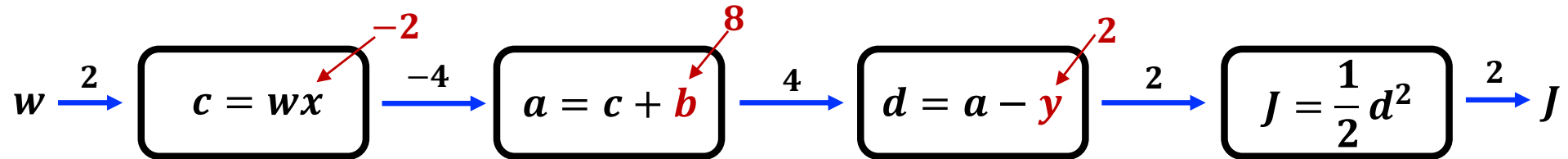
$$\frac{\partial J}{\partial d} = 2$$

$$d \uparrow \frac{0.001}{\epsilon} \quad J \uparrow \frac{0.002}{2\epsilon}$$

Computing the Derivatives

 **Forward prop**
 **Back prop**

$$w = 2 \quad b = 8 \quad x = -2 \quad y = 2 \quad a = wx + b \quad J(w, b) = \frac{1}{2} (a - y)^2$$



$$\frac{\partial J}{\partial a} = 2$$

$$\frac{\partial J}{\partial d} = 2$$

$$\begin{aligned}
 &a = 4.001 \quad d = 2.001 \\
 &\underline{a \uparrow 0.001 \quad d \uparrow 0.001} \\
 &\rightarrow J \uparrow 0.002
 \end{aligned}$$

$$d \uparrow \frac{0.001}{\epsilon} \quad J \uparrow \frac{0.002}{2\epsilon}$$

$$\frac{\partial J}{\partial a} = \frac{\partial d}{\partial a} \times \frac{\partial J}{\partial d}$$

2 1 2

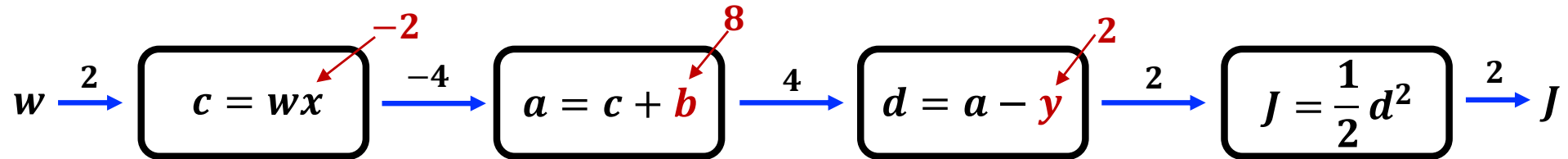
**Chain rule
(calculus)**



Computing the Derivatives

 **Forward prop**
 **Back prop**

$$w = 2 \quad b = 8 \quad x = -2 \quad y = 2 \quad a = wx + b \quad J(w, b) = \frac{1}{2} (a - y)^2$$



$$\frac{\partial J}{\partial c} = 2 \quad \frac{\partial J}{\partial b} = 2$$

$$c \uparrow 0.001 \quad a \uparrow 0.001 \rightarrow J \uparrow 0.002$$

$$\frac{\partial J}{\partial c} = \frac{\partial a}{\partial c} \times \frac{\partial J}{\partial a}$$

2 1 2

$$b \uparrow 0.001 \quad a \uparrow 0.001 \rightarrow J \uparrow 0.002$$

$$\frac{\partial J}{\partial b} = \frac{\partial a}{\partial b} \times \frac{\partial J}{\partial a}$$

2 1 2

$$\frac{\partial J}{\partial a} = 2$$

$$a = 4.001 \quad d = 2.001$$

$$\underline{a \uparrow 0.001 \quad d \uparrow 0.001} \rightarrow J \uparrow 0.002$$

$$\frac{\partial J}{\partial a} = \frac{\partial d}{\partial a} \times \frac{\partial J}{\partial d}$$

2 1 2

**Chain rule
(calculus)**

$$\frac{\partial J}{\partial d} = 2$$

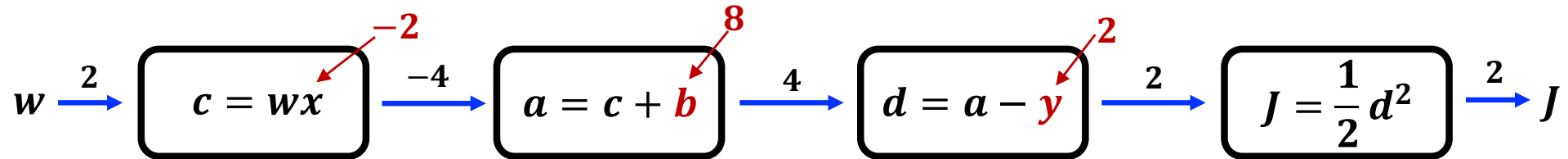
$$d \uparrow \frac{0.001}{\epsilon} \quad J \uparrow \frac{0.002}{2\epsilon}$$



Computing the Derivatives

 **Forward prop**
 **Back prop**

$$w = 2 \quad b = 8 \quad x = -2 \quad y = 2 \quad a = wx + b \quad J(w, b) = \frac{1}{2} (a - y)^2$$



$$\frac{\partial J}{\partial w} = -4$$

$$\frac{\partial J}{\partial c} = 2 \quad \frac{\partial J}{\partial b} = 2$$

$$\frac{\partial J}{\partial a} = 2$$

$$\frac{\partial J}{\partial d} = 2$$

$$\begin{aligned}
 w &= 2.001 & c &= -4.002 \\
 w \uparrow 0.001 & & c \downarrow 0.002 & \\
 & & c \uparrow -0.002 & \\
 & \rightarrow J \uparrow -0.004
 \end{aligned}$$

$$\frac{\partial J}{\partial w} = \frac{\partial c}{\partial w} \times \frac{\partial J}{\partial c}$$

$$\begin{array}{ccc}
 -4 & -2 & 2
 \end{array}$$

$$\begin{aligned}
 c \uparrow 0.001 & \quad a \uparrow 0.001 \\
 & \rightarrow J \uparrow 0.002
 \end{aligned}$$

$$\frac{\partial J}{\partial c} = \frac{\partial a}{\partial c} \times \frac{\partial J}{\partial a}$$

$$\begin{array}{ccc}
 2 & 1 & 2
 \end{array}$$

$$\begin{aligned}
 b \uparrow 0.001 & \quad a \uparrow 0.001 \\
 & \rightarrow J \uparrow 0.002
 \end{aligned}$$

$$\frac{\partial J}{\partial b} = \frac{\partial a}{\partial b} \times \frac{\partial J}{\partial a}$$

$$\begin{array}{ccc}
 2 & 1 & 2
 \end{array}$$

$$\begin{aligned}
 a &= 4.001 & d &= 2.001 \\
 a \uparrow 0.001 & \quad d \uparrow 0.001 \\
 & \rightarrow J \uparrow 0.002
 \end{aligned}$$

$$\frac{\partial J}{\partial a} = \frac{\partial d}{\partial a} \times \frac{\partial J}{\partial d}$$

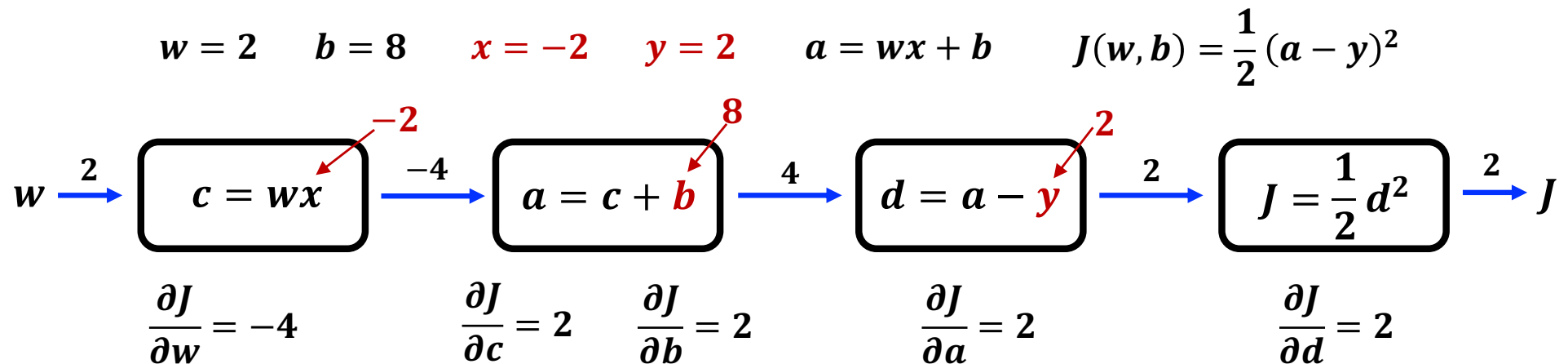
$$\begin{array}{ccc}
 2 & 1 & 2
 \end{array}$$

Chain rule (calculus)

$$\begin{aligned}
 d \uparrow \frac{0.001}{\epsilon} & \quad J \uparrow \frac{0.002}{2\epsilon}
 \end{aligned}$$



Computing the Derivatives



$$J = \frac{1}{2}((wx + b) - y)^2 = \frac{1}{2}((2 \times (-2) + 8) - 2)^2 = 2$$

$w \uparrow 0.001$

$$\rightarrow \frac{1}{2}((2.001 \times (-2) + 8) - 2)^2 = 1.996002 \approx 2 - 0.004$$

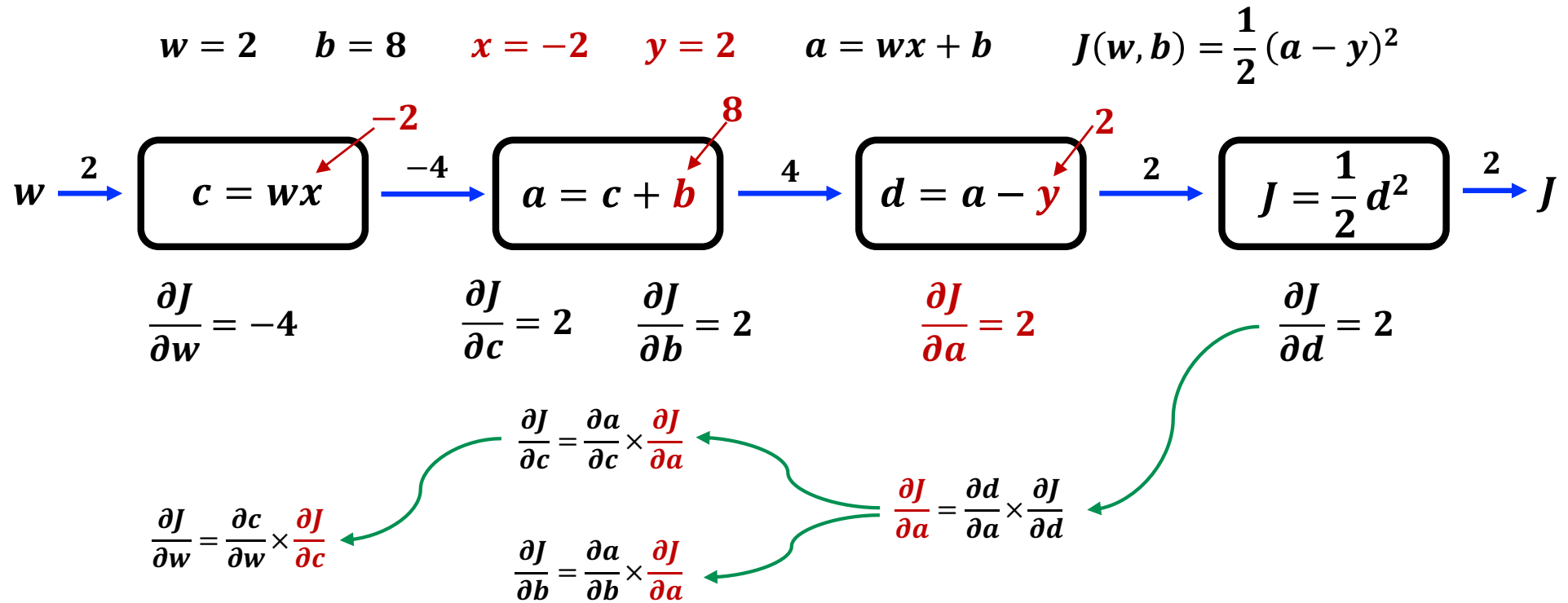
$$J \downarrow 4 \times 0.001$$

$$J \uparrow -4 \times 0.001$$

$$\frac{\partial J}{\partial w} = -4$$



Backprop is an Efficient Way



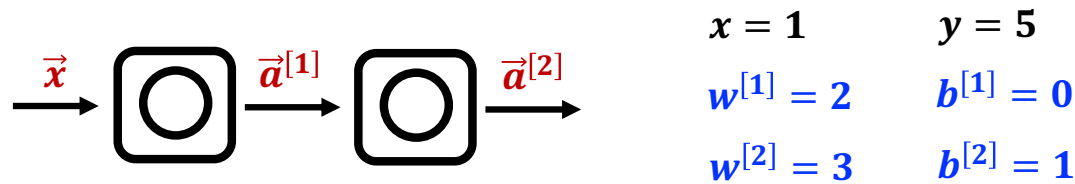
Compute $\frac{\partial J}{\partial a}$ once and use it to compute both $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$

If N nodes and P parameters, compute derivatives in roughly $N+P$ steps rather than $N \times P$ steps.

N	P	N+P	$N \times P$
10^4	10^5	1.1×10^5	10^9

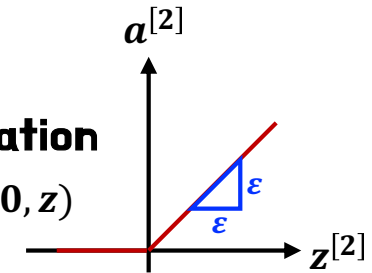


Neural Network Example



ReLU activation

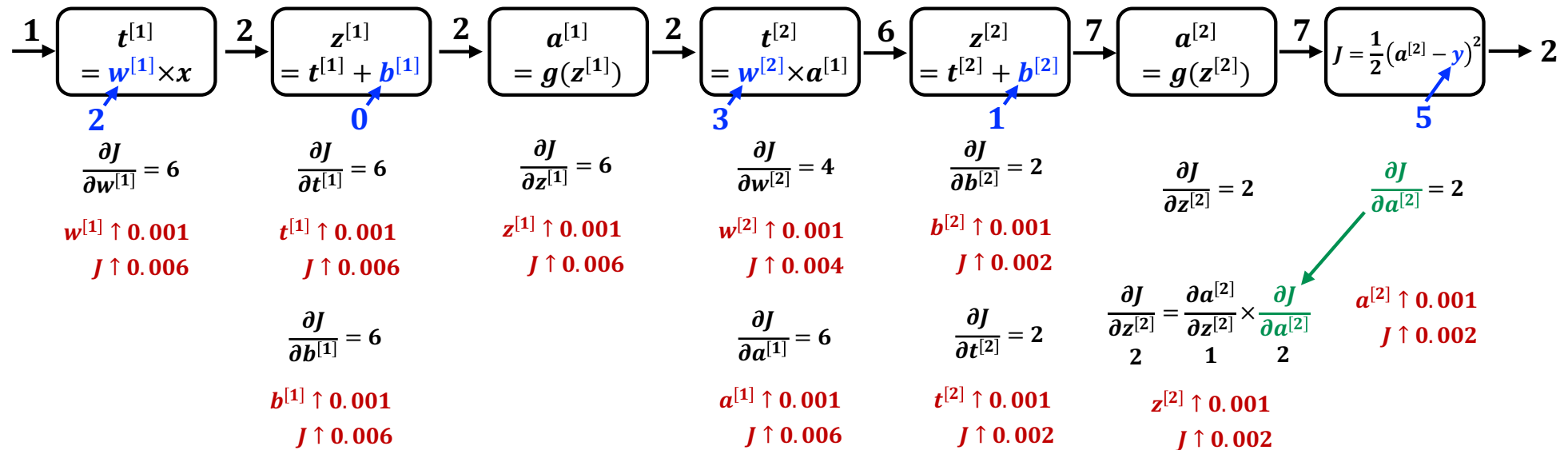
$$g(z) = \max(0, z)$$



$$a^{[1]} = g(z^{[1]}) = g(w^{[1]}x + b^{[1]}) = g(2 \times 1 + 0) = g(2) = 2$$

$$a^{[2]} = g(z^{[2]}) = g(w^{[2]}a^{[1]} + b^{[2]}) = g(3 \times 2 + 1) = g(7) = 7$$

$$J(w, b) = \frac{1}{2}(a^{[2]} - y)^2 = \frac{1}{2}(7 - 5)^2 = 2$$



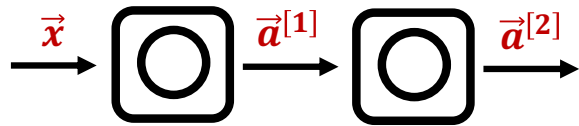
$$\frac{\partial J}{\partial w^{[1]}} = 6$$

Backpropagation

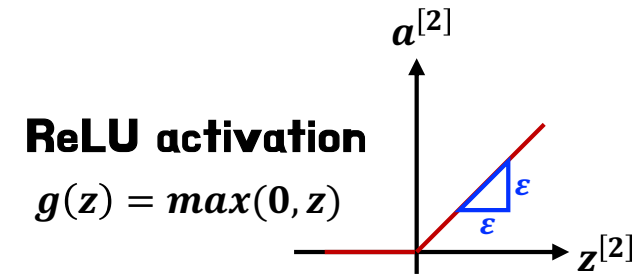
If $w^{[1]} \uparrow \epsilon$, then $J \uparrow 6 \times \epsilon$. Let's verify this!



Neural Network Example



$$\begin{aligned} x &= 1 & y &= 5 \\ w^{[1]} &= 2 & b^{[1]} &= 0 \\ w^{[2]} &= 3 & b^{[2]} &= 1 \end{aligned}$$



$$a^{[1]} = g(z^{[1]}) = g(w^{[1]}x + b^{[1]}) = g(2 \times 1 + 0) = g(2) = 2$$

$$a^{[2]} = g(z^{[2]}) = g(w^{[2]}a^{[1]} + b^{[2]}) = g(3 \times 2 + 1) = g(7) = 7$$

$$J(w, b) = \frac{1}{2} (a^{[2]} - y)^2 = \frac{1}{2} (7 - 5)^2 = 2$$

$$a^{[1]} = g(z^{[1]}) = g(w^{[1]}x + b^{[1]}) = g(2.001 \times 1 + 0) = g(2.001) = 2.001$$

$$a^{[2]} = g(z^{[2]}) = g(w^{[2]}a^{[1]} + b^{[2]}) = g(3 \times 2.001 + 1) = g(7.003) = 7.003$$

$$J(w, b) = \frac{1}{2} (a^{[2]} - y)^2 = \frac{1}{2} (7.003 - 5)^2 = 2.006005$$

$$\begin{aligned} w^{[1]} &\uparrow 0.001 & \frac{\partial J}{\partial w^{[1]}} &= 6 \\ J &\uparrow 0.006 \end{aligned}$$

Evaluating model

Debugging a Learning Algorithm

You've implemented regularized linear regression on housing prices

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

**But it makes unacceptably large errors in predictions.
What do you try next?**

- **Get more training examples**
- **Try smaller sets of features**
- **Try getting additional features**
- **Try adding polynomial features ($x_1^2, x_2^2, x_1x_2, etc$)**
- **Try decreasing λ**
- **Try increasing λ**



Machine Learning Diagnostic

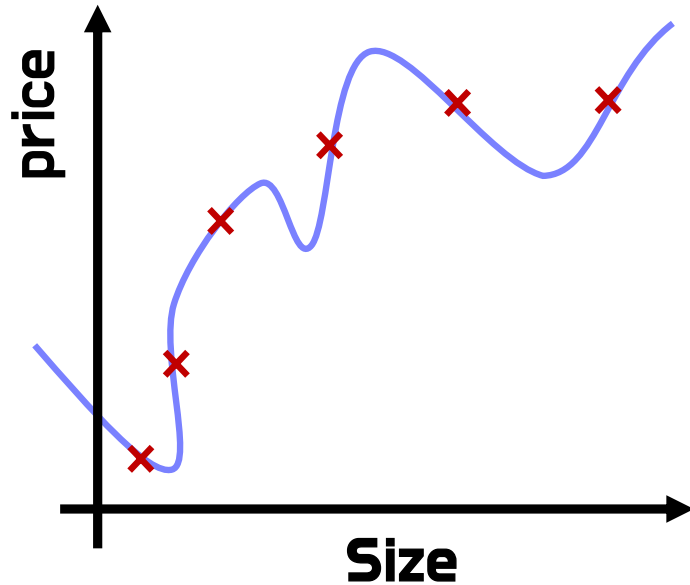
Diagnostic:

**A test that you run to gain insight into what is/isn't working
With a learning algorithm, to gain guidance into improving its
performance**

Diagnostics can take time to implement

But It will be useful to improve your model properly

Evaluating Your Model



**Model fits the training data well,
but will fail to generalize to new
examples not in the training set**

→ Need the way to evaluate model

$$f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + \dots + w_nx^n + b$$

- x_1 = size in feet²
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of home in years

Evaluating Your Model

Dataset:

	size	price
	2200	390
	1700	325
	2450	365
	1400	220
	3100	536
70 %	1930	300
	1530	320
	1430	188
30 %	1375	210
	1485	235

Training set

m_{train} = no. training examples

$$\begin{matrix} (x^{(1)}, y^{(1)}) \\ (x^{(2)}, y^{(2)}) \\ \vdots \\ (x^{(m_{train})}, y^{(m_{train})}) \end{matrix}$$

Test set

m_{test} = no. test examples

$$\begin{matrix} (x^{(1)}, y^{(1)}) \\ \vdots \\ (x^{(m_{test})}, y^{(m_{test})}) \end{matrix}$$



Train/Test Procedure for Linear Regression

Fit parameters by minimizing cost function $J(\vec{w}, b)$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

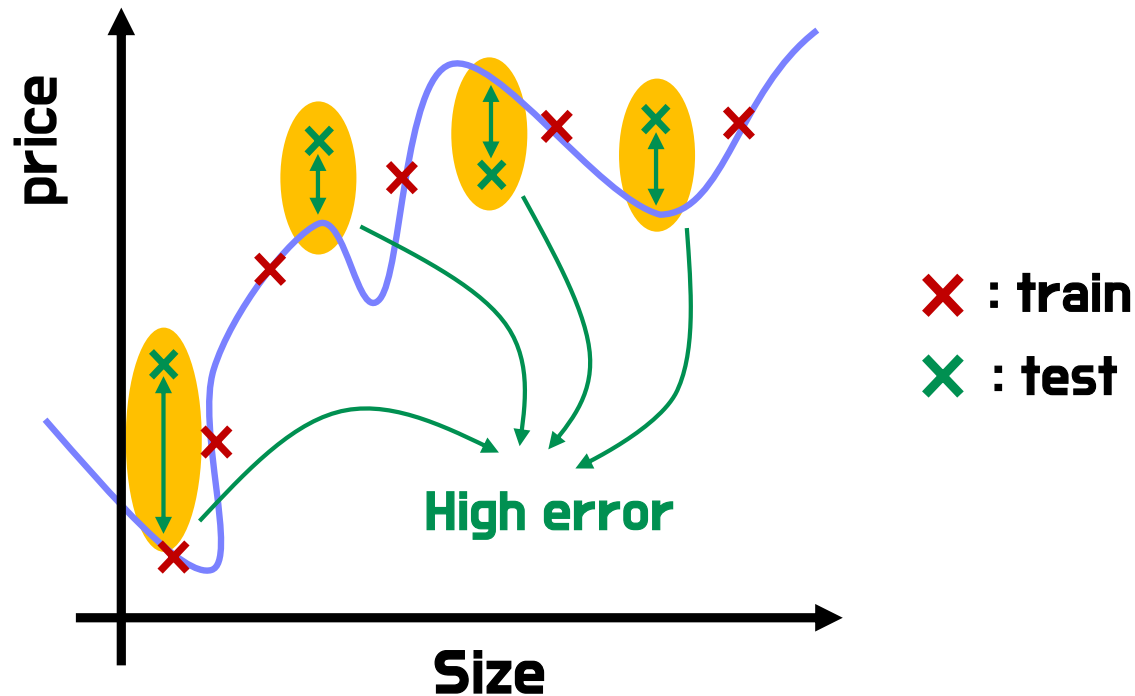
Compute test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (f_{\vec{w}, b}(\vec{x}_{test}^{(i)}) - y_{test}^{(i)})^2$$

Compute training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} (f_{\vec{w}, b}(\vec{x}_{train}^{(i)}) - y_{train}^{(i)})^2$$

Train/Test Procedure for Linear Regression



$J_{train}(\vec{w}, b)$ will be low $J_{test}(\vec{w}, b)$ will be high

Train/Test Procedure for Classification

Fit parameters by minimizing $J(\vec{w}, b)$ to find $\vec{w}, b$

For example,

$$J(\vec{w}, b) = \frac{-1}{m_{train}} \sum_{i=1}^{m_{train}} y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)})) + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

Compute test error:

$$J_{test}(\vec{w}, b) = \frac{-1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log(f_{\vec{w}, b}(\vec{x}_{test}^{(i)})) + (1 - y_{test}^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}_{test}^{(i)}))$$

Compute training error:

$$J_{train}(\vec{w}, b) = \frac{-1}{m_{train}} \sum_{i=1}^{m_{train}} y_{train}^{(i)} \log(f_{\vec{w}, b}(\vec{x}_{train}^{(i)})) + (1 - y_{train}^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}_{train}^{(i)}))$$



Train/Test Procedure for Classification

Fraction of the test set and the fraction of the train set that the algorithm has misclassified.

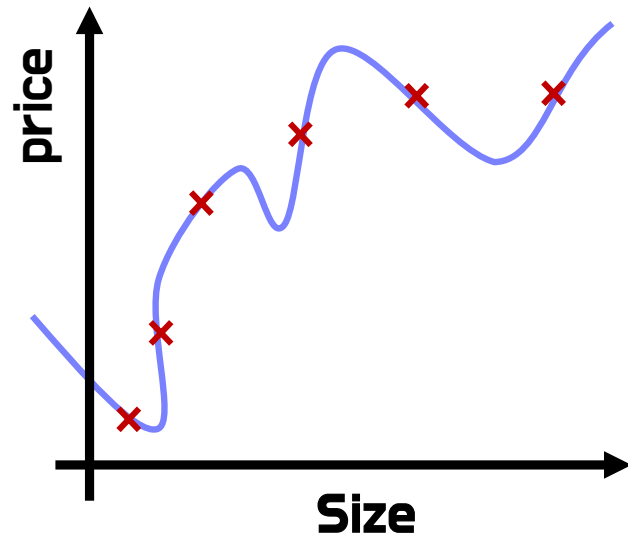
$$\hat{y} = \begin{cases} 1 & \text{if } f_{\vec{w},b}(\vec{x}^{(i)}) \geq 0.5 \\ 0 & \text{if } f_{\vec{w},b}(\vec{x}^{(i)}) < 0.5 \end{cases}$$

Count $\hat{y} \neq y$

$J_{test}(\vec{w}, b)$ is the fraction of the test set that has been misclassified.

$J_{train}(\vec{w}, b)$ is the fraction of the train set that has been misclassified.

Model Selection



Once parameters $\vec{w}, b$ are fit to the training set, the training error $J_{train}(\vec{w}, b)$ is likely lower than the actual generalization error.

$J_{test}(\vec{w}, b)$ is better estimate of how well the model will generalize to new data compared to $J_{train}(\vec{w}, b)$.

$$x = x_1$$

$$f_{\vec{w}, b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$$

Model Selection

$d = 1$	1. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + b$	$\longrightarrow$	$w^{<1>}, b^{<1>}$	$J_{test}(w^{<1>}, b^{<1>})$
$d = 2$	2. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + b$	$\longrightarrow$	$w^{<2>}, b^{<2>}$	$J_{test}(w^{<2>}, b^{<2>})$
$d = 3$	3. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + w_3x^3 + b$	$\longrightarrow$	$w^{<3>}, b^{<3>}$	$J_{test}(w^{<3>}, b^{<3>})$
	$\vdots$			
$d = 10$	10. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + \dots + w_{10}x^{10} + b$	$\longrightarrow$	$w^{<10>}, b^{<10>}$	$J_{test}(w^{<10>}, b^{<10>})$

Choose $w_1x + \dots + w_5x^5 + b$ $d = 5$ $J_{test}(w^{<5>}, b^{<5>})$

How well does the model perform ?

- **Report test set error** $J_{test}(w^{<5>}, b^{<5>})$?
- **The problem:** $J_{test}(w^{<5>}, b^{<5>})$ **is too optimistic.**
- **Since extra parameter d (degree of polynomial) was chosen**
- **That is, test set is spoiled**
- **Need the different way !**

Training/Cross Validation/Test

Dataset:

	size	price
60 %	2200	390
	1700	325
	2450	365
	1400	220
	3100	536
	1930	300
20 %	1530	320
	1430	188
20 %	1375	210
	1485	235

Training set

m_{train} = no. training examples

$$\begin{matrix} (x^{(1)}, y^{(1)}) \\ (x^{(2)}, y^{(2)}) \\ \vdots \\ (x^{(m_{train})}, y^{(m_{train})}) \end{matrix}$$

Cross validation

m_{cv} = no. cross validation examples

$$\begin{matrix} (x^{(1)}, y^{(1)}) \\ \vdots \\ (x^{(m_{cv})}, y^{(m_{cv})}) \end{matrix}$$

Test set

m_{test} = no. test examples

$$\begin{matrix} (x^{(1)}, y^{(1)}) \\ \vdots \\ (x^{(m_{test})}, y^{(m_{test})}) \end{matrix}$$



Training/Cross Validation/Test

Training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(f_{\vec{w}, b}(\vec{x}_{train}^{(i)}) - y_{train}^{(i)} \right)^2$$

Cross validation error:

$$J_{CV}(\vec{w}, b) = \frac{1}{2m_{CV}} \sum_{i=1}^{m_{CV}} \left(f_{\vec{w}, b}(\vec{x}_{CV}^{(i)}) - y_{CV}^{(i)} \right)^2$$

Test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(f_{\vec{w}, b}(\vec{x}_{test}^{(i)}) - y_{test}^{(i)} \right)^2$$



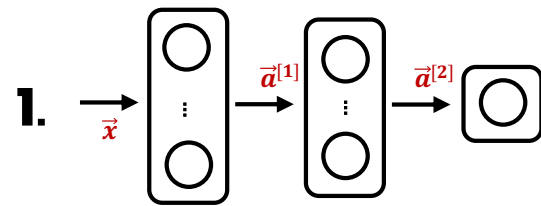
Model Selection

$d = 1$	1. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + b$	$\longrightarrow$	$w^{<1>}, b^{<1>}$	$J_{CV}(w^{<1>}, b^{<1>})$
$d = 2$	2. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + b$	$\longrightarrow$	$w^{<2>}, b^{<2>}$	$J_{CV}(w^{<2>}, b^{<2>})$
$d = 3$	3. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + w_3x^3 + b$	$\longrightarrow$	$w^{<3>}, b^{<3>}$	$J_{CV}(w^{<3>}, b^{<3>})$
	$\vdots$			
$d = 10$	10. $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + w_2x^2 + \dots + w_{10}x^{10} + b$	$\longrightarrow$	$w^{<10>}, b^{<10>}$	$J_{CV}(w^{<10>}, b^{<10>})$

Pick $f_{\vec{w},b}(\vec{x}^{(i)}) = w_1x + \dots + w_4x^4 + b$ $J_{CV}(w^{<4>}, b^{<4>})$

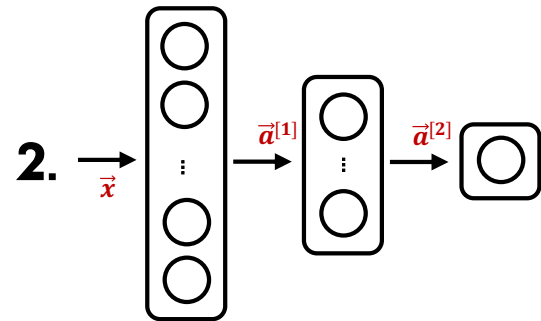
Estimate generalization error using test set $J_{test}(w^{<4>}, b^{<4>})$

Model Selection



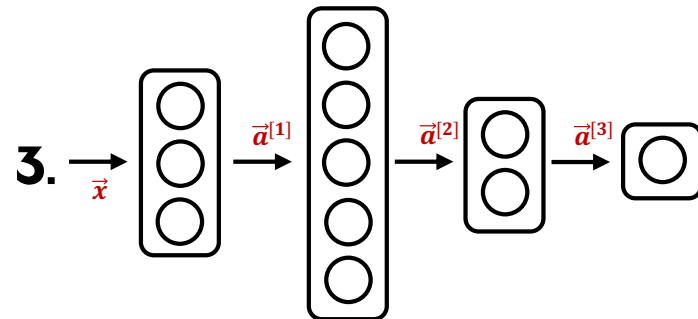
$$w^{<1>}, b^{<1>}$$

$$J_{CV}(w^{<1>}, b^{<1>})$$



$$w^{<2>}, b^{<2>}$$

$$J_{CV}(w^{<2>}, b^{<2>})$$



$$w^{<3>}, b^{<3>}$$

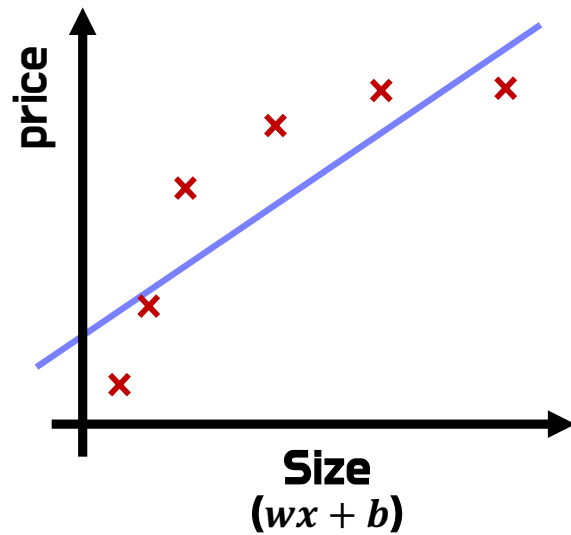
$$J_{CV}(w^{<3>}, b^{<3>})$$

Pick $w^{<2>}, b^{<2>}$ $J_{CV}(w^{<2>}, b^{<2>})$

Estimate generalization error using the test set $J_{test}(w^{<2>}, b^{<2>})$

Bias and variance

Bias/Variance

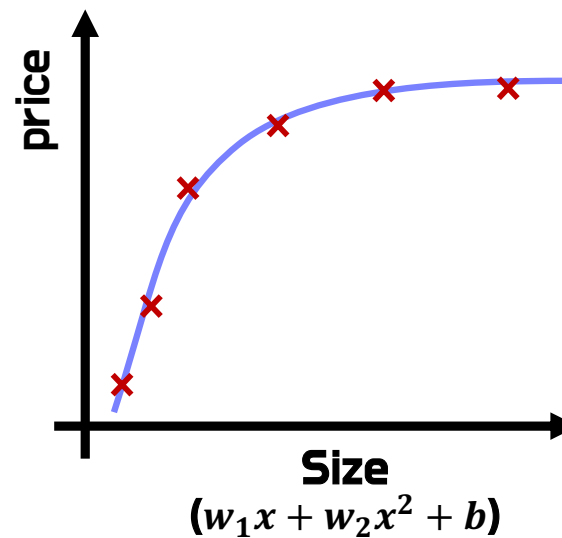


High bias
(Underfit)

J_{train} is high

J_{CV} is high

$$d = 1$$

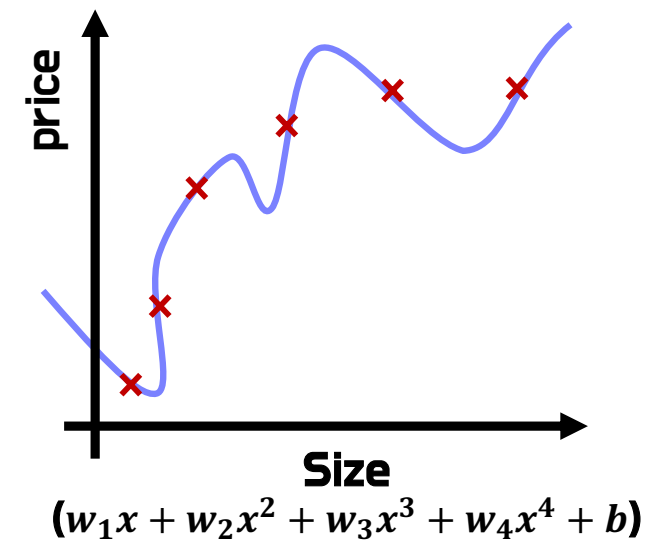


Just right

J_{train} is low

J_{CV} is low

$$d = 2$$



High variance
(Overfit)

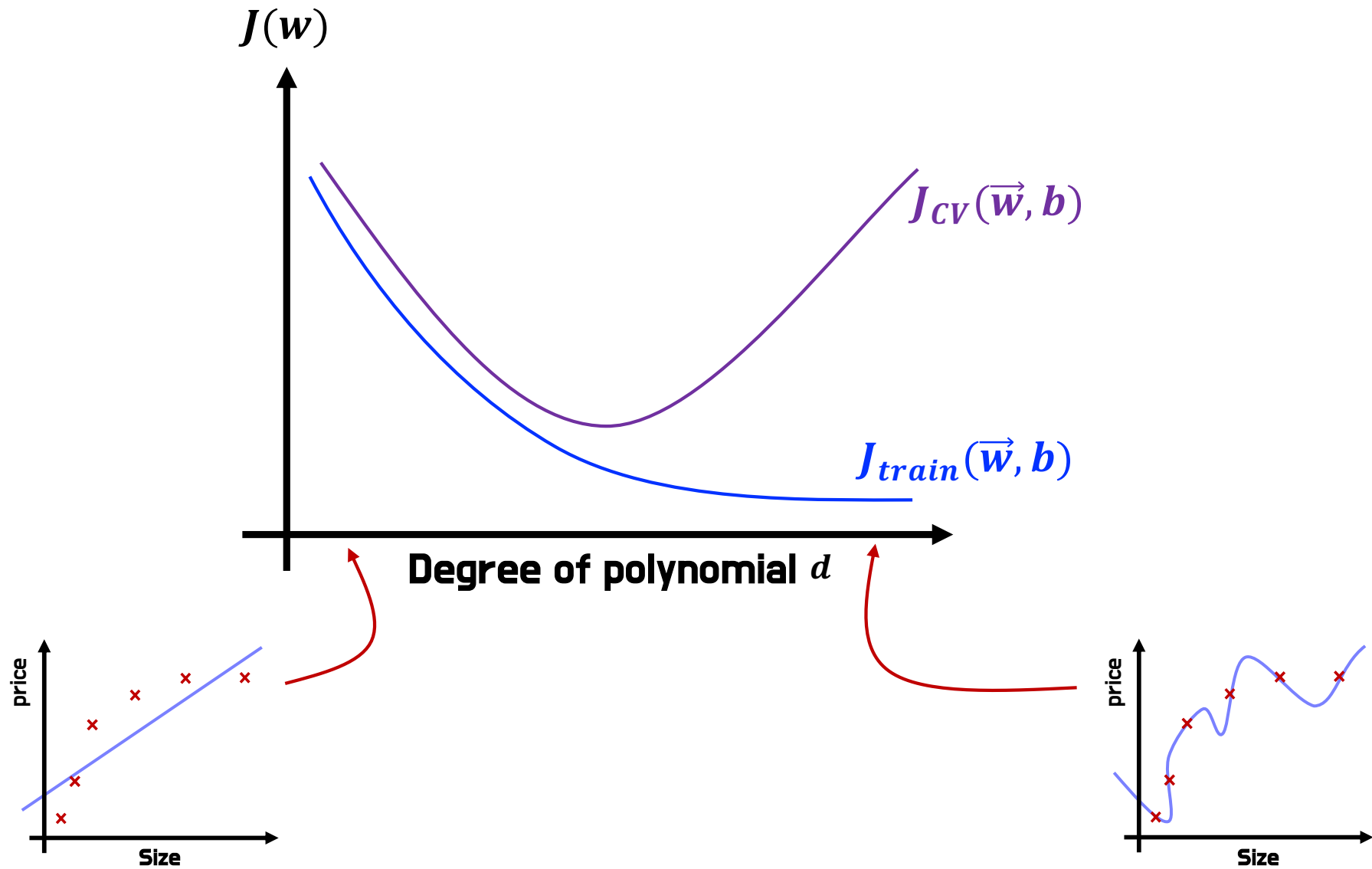
J_{train} is low

J_{CV} is high

$$d = 4$$

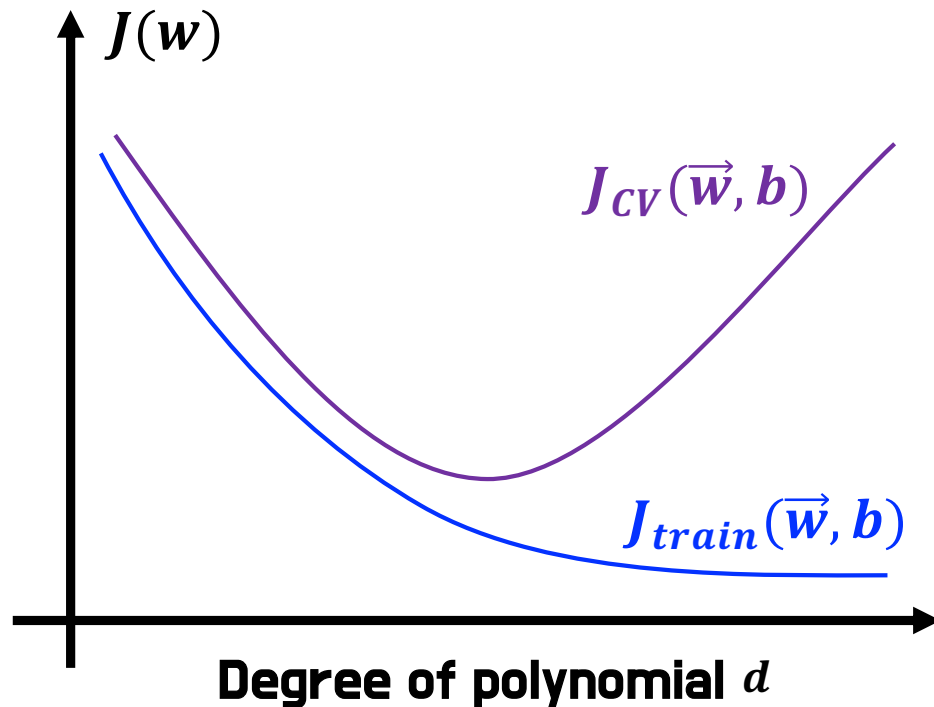


Understanding Bias and Variance



Diagnosing Bias and Variance

How do you tell if your algorithm has a bias or variance problem ?



High bias (**underfit**)

J_{train} is high

$J_{train} \sim J_{cv}$

High variance (**overfit**)

$J_{cv} \gg J_{train}$

J_{train} may be low

High bias and high variance

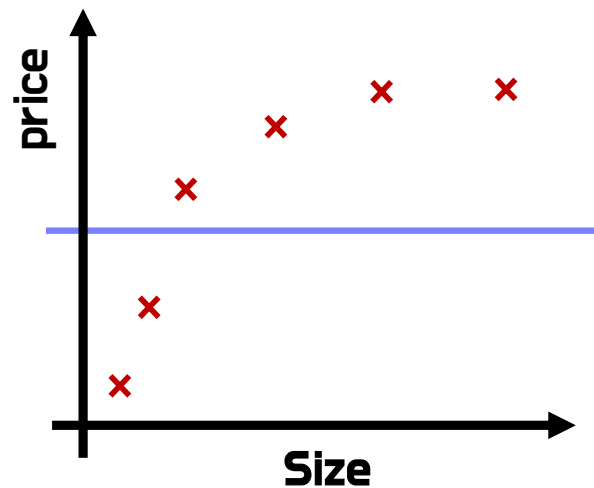
J_{train} will be high

$J_{cv} \gg J_{train}$

Linear Regression with Regularization

Model: $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

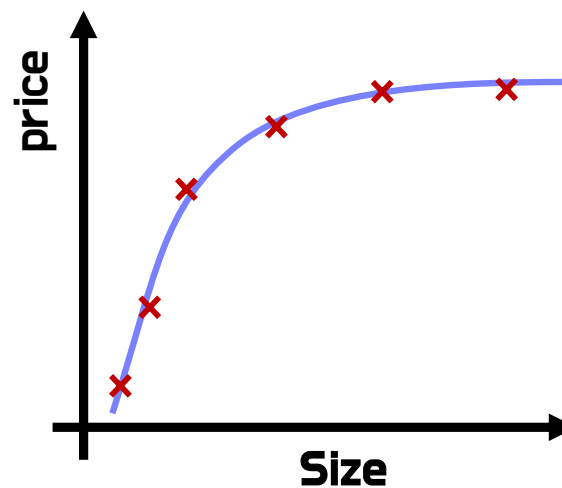


Large λ

High bias (underfit)

J_{train} is high

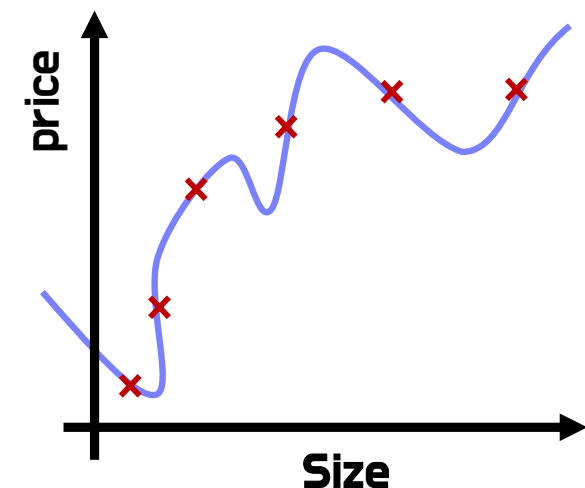
J_{CV} is high



Intermediate λ

J_{train} is low

J_{CV} is low



Small λ

High variance (overfit)

J_{train} is low

J_{CV} is high



Choosing the Regularization Parameter

Model: $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

1. Try $\lambda = 0$	→	$w^{<1>}, b^{<1>}$	$J_{CV}(w^{<1>}, b^{<1>})$
2. Try $\lambda = 0.01$	→	$w^{<2>}, b^{<2>}$	$J_{CV}(w^{<2>}, b^{<2>})$
3. Try $\lambda = 0.02$	$\min_{\vec{w}, b} J(\vec{w}, b)$	$w^{<3>}, b^{<3>}$	$J_{CV}(w^{<3>}, b^{<3>})$
4. Try $\lambda = 0.04$		$w^{<4>}, b^{<4>}$	$J_{CV}(w^{<4>}, b^{<4>})$
5. Try $\lambda = 0.08$	→	$w^{<5>}, b^{<5>}$	$J_{CV}(w^{<5>}, b^{<5>})$
⋮			
12. Try $\lambda \approx 10$	→	$w^{<12>}, b^{<12>}$	$J_{CV}(w^{<12>}, b^{<12>})$

Pick $w^{<5>}, b^{<5>}$ $J_{CV}(w^{<5>}, b^{<5>})$

Report test error: $J_{test}(w^{<5>}, b^{<5>})$



Bias and Variance as Lambda

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^m (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n w_j^2$$

