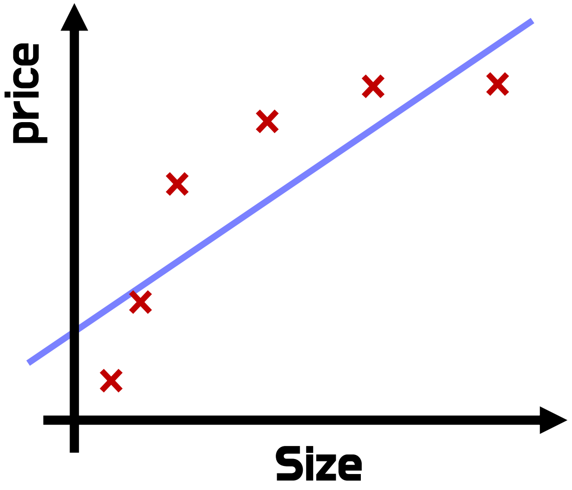
Machine Learning 04

Kihyun Shin
DMSE, HBNU

TheMAD

Overfitting

Regression example



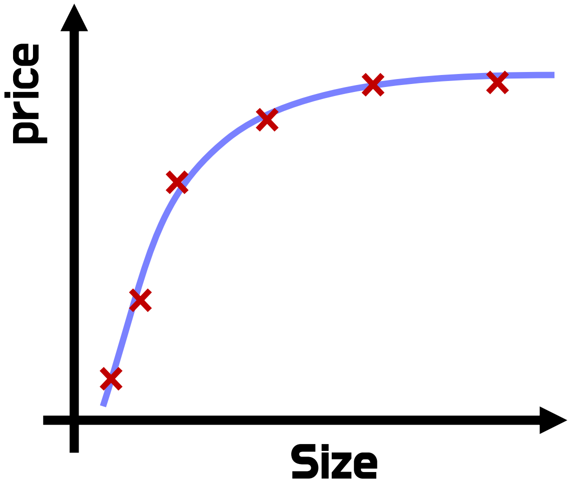
$(wx + b)$

Underfit

Does not fit the training set well

High bias

Food is too cold



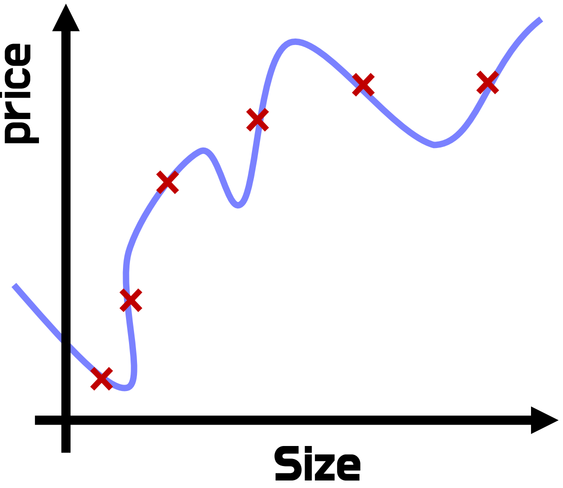
$(w_1x + w_2x^2 + b)$

Just right

Fits training set pretty well

Generalization

Food is warm



$(w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b)$

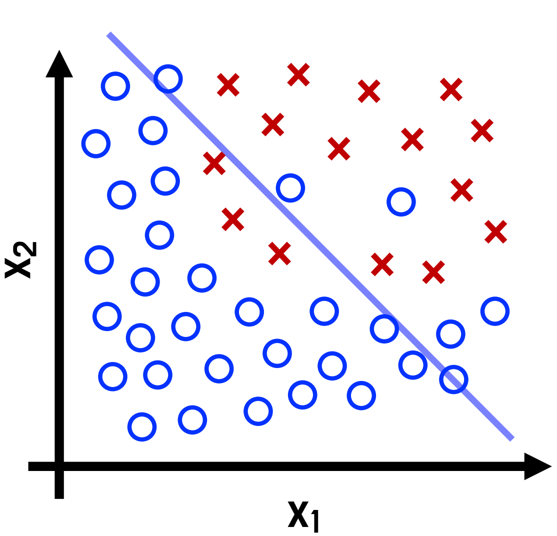
Overfit

Fits the training set extremely well

High variance

Food is too hot

Classification

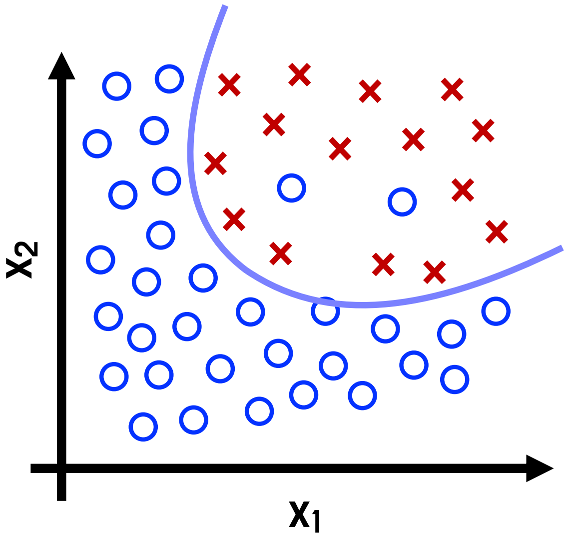


$$z = w_1x_1 + w_2x_2 + b$$

$$f_{w,b}(x) = g(z)$$

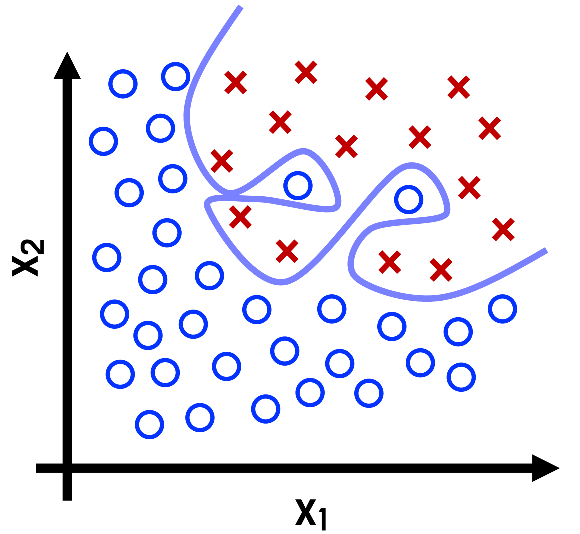
g is the sigmoid function

Underfit



$$z = w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2 + b$$

Just right



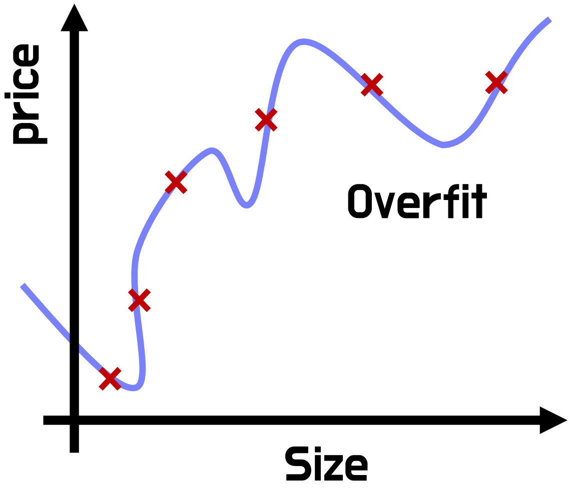
$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + w_6x_1^3x_2 + \ldots + b$$

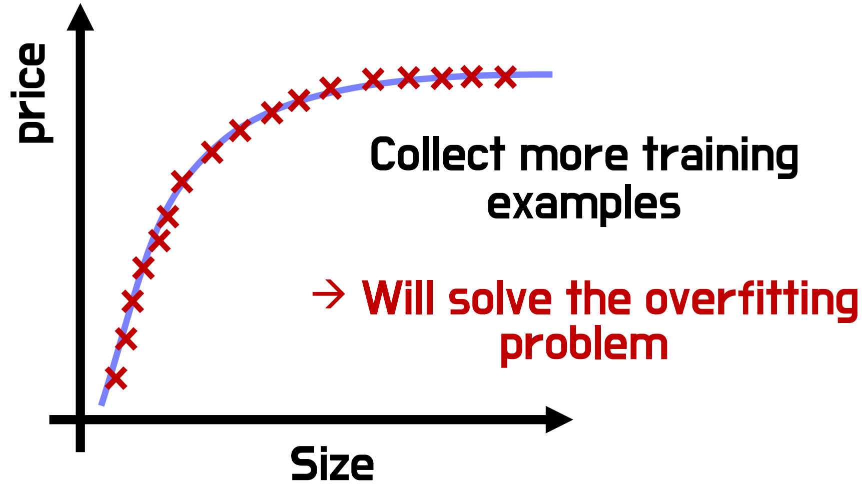
Overfit

How to overcome overfit

1. More data
2. Feature selection
3. Regularization

1. Collect more training examples





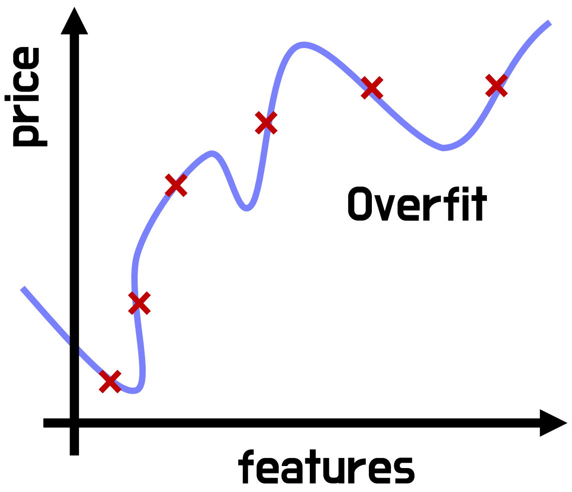
2. Select Features to include/exclude

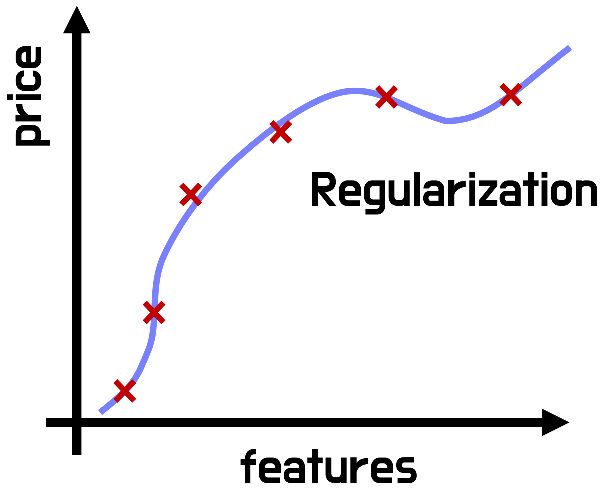
Size x_1	Bedrooms x_2	Floors x_3	Age x_4	Avg. income x_5	…	Distance to coffee shop x_{100}	Price y

⋮

All features	Selected features
with insufficient data	1. Size 2. Bedrooms 3. Age "Feature selection" → Useful features could be lost
Overfit	Just right

3. Regularization





$$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$$

Large value for w_j

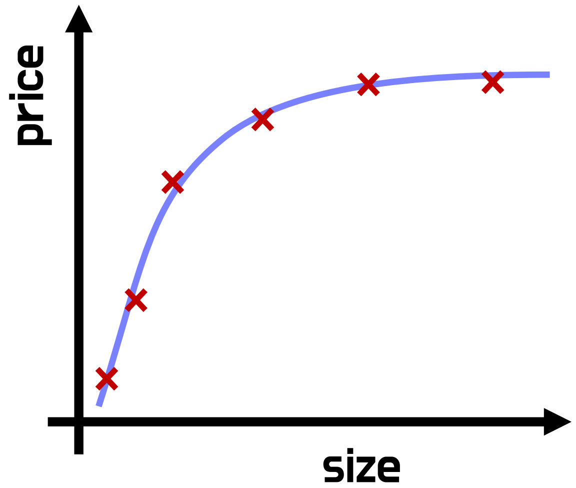
$$f(x) = 13x - 0.23x^2 + 0.000014x^3 - 0.0001x^4 + 10$$

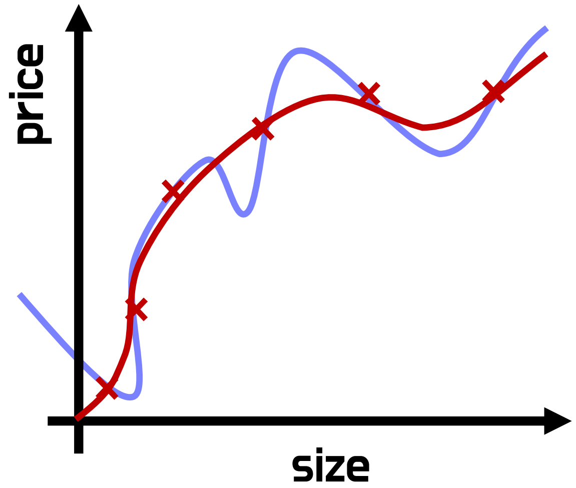
Small value for w_j

cf.) $f(x)x = 28x - 385x^2 + 39x^3 - \cancelto{0}{174x^4}$

"Feature selection"

3. Regularization - Intuition





$$w_1 x + w_2 x^2 + b$$

$$w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

$$w_1 x + w_2 x^2 + \cancel{w_3 x^3}^{\,0} + \cancel{w_4 x^4}^{\,0} + b$$

To make w_3, w_4 really small (~ 0)

$$\min_{\vec{w},b} \frac{1}{2m} \sum_{i=1}^{m} (f_{\vec{w},b}(x^{(i)}) - y^{(i)})^2 + 1000 \underset{0.001}{w_3^2} + 1000 \underset{0.002}{w_4^2}$$

TheMAD

3. Regularization

Small values $w_1, w_2, w_3, \dots, w_n, b$ $\longrightarrow$ Simpler model
Less likely to overfit ($w_3 \sim 0, w_4 \sim 0$)

Size x_1	Bedrooms x_2	Floors x_3	Age x_4	Avg. income x_5	...	Distance to coffee shop x_{100}	Price y

⋮

$w_1, w_2, w_3, \dots, w_n, b$

n **features** ($n = 100$)

"lambda" regularization parameter ($\lambda > 0$)

Regularization term

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

General cost function

Working in different size of training sets

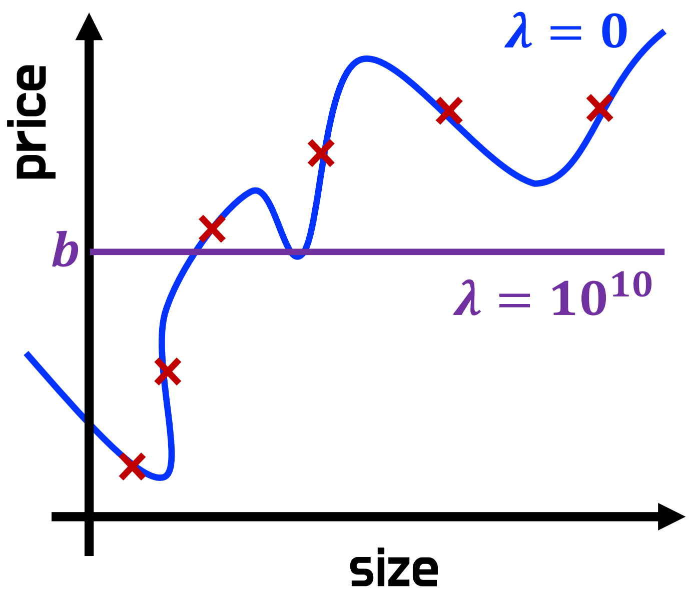
can be include (or exclude)

 TheMAD

3. Regularization

Mean squared error — Regularization term

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



If $\lambda = 0$,

No regularization → overfit

If $\lambda = 10^{10}$,

$$f_{\vec{w},b}(\vec{x}) = \cancelto{0}{w_1 x} + \cancelto{0}{w_2 x^2} + \cancelto{0}{w_3 x^3} + \cancelto{0}{w_4 x^4} + b$$
$$= b$$

- λ balances both goals

  (fit the data and keep w_j small)

- Choose λ wisely

Regularization
(Linear regression)

Regularized linear regression

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

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$$

Don't have to regularize b

Implementing gradient descent

Repeat

$$w_j = w_j - \alpha[\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m} w_j]$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$ → **Simultaneous update**

$$w_j = w_j - \underbrace{\alpha\frac{\lambda}{m}w_j}_{w_j(1-\alpha\frac{\lambda}{m})} - \underbrace{\alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_j^{(i)}}_{\text{Usual update}}$$

$w_j(1 - \alpha\frac{\lambda}{m})$

Shrink w_j

Usual update

For example)

$$\alpha\frac{\lambda}{m} = 0.01\frac{1}{50} = 0.0002$$

$$w_j\underbrace{(1 - 0.0002)}_{0.9998}$$

How we get the derivative term

$$\frac{\partial}{\partial w_j} J(w_j, b) = \frac{\partial}{\partial w_j}\left[\frac{1}{2m}\sum_{i=1}^{m}\left(f\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2\right]$$

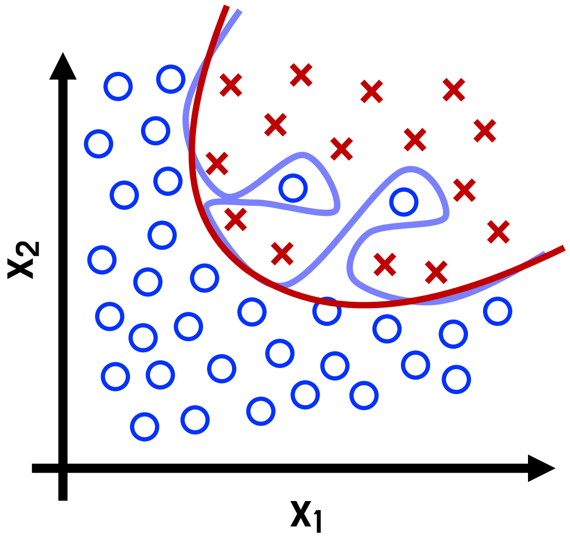
$$= \frac{1}{\cancel{2}m}\sum_{i=1}^{m}\left(\vec{w}\cdot\vec{x}^{(i)} + b - y^{(i)}\right)\cancel{2}x^{(i)} + \frac{\lambda}{\cancel{2}m}\cancel{2}w_j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(\underbrace{\vec{w}\cdot\vec{x}^{(i)} + b}_{f\left(\vec{x}^{(i)}\right)} - y^{(i)}\right)x^{(i)} + \frac{\lambda}{m}w_j$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(f\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x^{(i)} + \frac{\lambda}{m}w_j$$

Regularization
(Logistic regression)

Regularized logistic regression



$$z = w_1x_1 + w_2x_2 + w_3x_1^2x_2 + w_4x_1^2x_2^2 + w_5x_1^2x_2^3 + w_6x_1^3x_2 + \ \dots + \ b$$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-z}}$$

Cost function

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

$\min_{\vec{w},b} J(\vec{w},b)$ and $w_j \downarrow$

Regularized logistic regression

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

Looks identical with linear regression

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w},b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m} w_j$$

Logistic regression

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w},b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$$

Don't have to regularize b

Multiple linear regression

Overview - one feature

One feature →

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	400
1416	232
1534	315
852	178
…	…

$$f_{w,b}(x) = wx + b$$

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f(x^{(i)}) - y^{(i)})^2$$

Multiple features (variables)

Size in feet² x_1	Number of bedrooms x_2	Number of floors x_3	Age of home in years x_4	Price ($) in $1000's y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

$x_j = j^{th}$ feature $\quad j = 1 \ldots 4$

$n =$ number of features $\quad n = 4$

$\vec{x}^{(i)} =$ features of i^{th} training example $\quad \vec{x}^{(2)} = [1416, 3, 2, 40,]$

${x_j}^{(i)} =$ value of j^{th} feature in i^{th} training example $\quad {x_2}^{(2)} = 3$

Model:

Previously (one feature): $f_{w,b}(x) = wx + b$

Now,

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

For example,

$$f_{w,b}(x) = 0.1\underbrace{x_1}_{\text{size}} + 4\underbrace{x_2}_{\text{\# of bedrooms}} + 10\underbrace{x_3}_{\text{\# of floors}} - 2\underbrace{x_4}_{\text{years}} + \underbrace{80}_{\text{base price}}$$

Model:

$$f_{w,b}(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$\vec{w} = [w_1, w_2, w_3, w_4]$

b is a number

$\vec{x} = [x_1, x_2, x_3, x_4]$

vector

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

Dot product

Multiple linear regression

Vectorization
(for the efficiency)

Vectorization

- **Parameters and features**

$$\vec{w} = [w_1, w_2, w_3]$$

b is a number

$$\vec{x} = [x_1, x_2, x_3]$$

Linear algebra: count from 1

$w[0]$ $w[1]$ $w[2]$

```
w = np.array([1.0, 2.5, -3.3])
b = 4
x = np.array([10, 20, 30])
```

$x[0]$ $x[1]$ $x[2]$

code: count from 0

- **Without vectorization**

$$f_{\vec{w},b}(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

```
f = w[0] * x[0] +
    w[1] * x[1] +
    w[2] * x[2] + b
```

$$f_{\vec{w},b}(\vec{x}) = \sum_{j=1}^{n} w_j x_j + b$$

```
f = 0
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
```

- **With vectorization**

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

Short coding and fast calculation →

```
f = np.dot(w,x) + b
```

Vectorization

- **Without vectorization**

```
for j in range(0,16):
    f = f + w[j] * x[j]
```

t_0 f + w[0] * x[0]

t_1 f + w[1] * x[1]

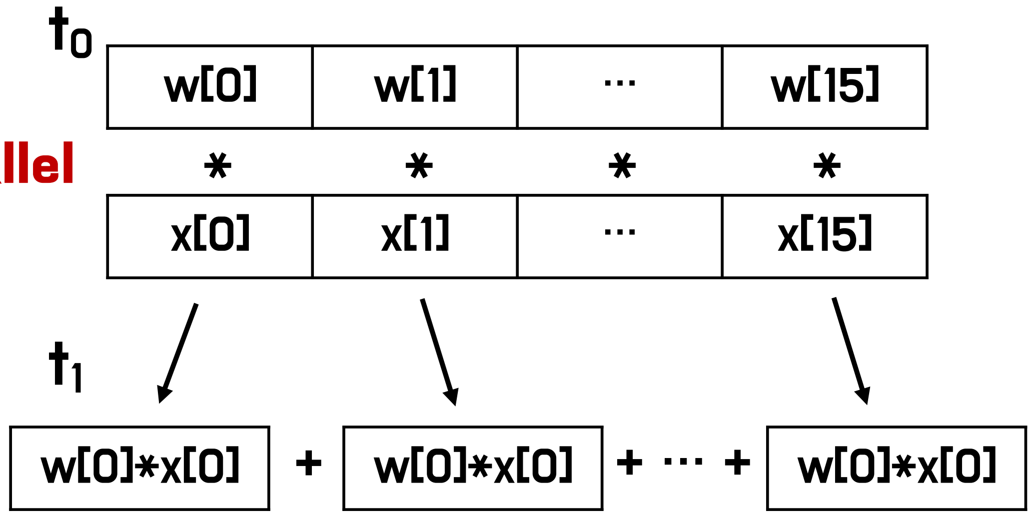
…

t_{15} f + w[15] * x[15]

- **With vectorization**

```
np.dot(w,x)
```

In parallel



efficient → scale to large datasets

Vectorization for gradient descent

$\vec{w} = [w_1, w_2, \dots, w_{16}]$ $\cancel{b}$

$\vec{d} = [d_1, d_2, \dots, d_{16}]$ ⟶ **derivatives**

```
w = np.array([0.5, 1.3, … , 3.4])
d = np.array([0.3, 0.2, … , 0.4])
```

$w_j = w_j - \underbrace{0.1}_{\text{Learning rate } \alpha} d_j \; for \; j = 1 \; \dots 16$

- **Without vectorization**

$w_1 = w_1 - 0.1 d_1$

$w_2 = w_2 - 0.1 d_2$

$\dots$

$w_{16} = w_{16} - 0.1 d_{16}$

```
for j in range(0,16):
    w[j] = w[j] – 0.1*d[j]
```

- **With vectorization**

$\vec{w} = \vec{w} - 0.1\vec{d}$

In parallel

w[0]	w[1]	…	w[16]
−	−	−	−
0.1*d[0]	0.1*d[1]	…	0.1*d [15]
↓	↓	↓	↓
w[0]	w[1]	…	w[16]

```
w = w – 0.1*d
```

Notation comparison

	Previous notation	Vector notation
Parameters	$w_1, \ldots, w_n \quad b$	$\vec{w} = [w_1, \ldots, w_n] \quad b$
Model	$f_{\vec{w},b}(\vec{x}) = w_1x_1 + \cdots + w_nx_n + b$	$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$
Cost function	$J(w_1, \ldots, w_n, b)$	$J(\vec{w}, b)$
Gradient descent	$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \ldots, w_n, b)$	$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$
	$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \ldots, w_n, b)$	$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$

Gradient descent

One feature

$$w = w - \alpha[\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}]$$

$$\frac{\partial}{\partial w}J(w,b)$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

Simultaneously update w, b

n features ($n \geq 2$)

$$w_1 = w_1 - \alpha[\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_1^{(i)}]$$

$$\frac{\partial}{\partial w_1}J(\vec{w},b)$$

…

$$w_n = w_n - \alpha[\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})x_n^{(i)}]$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})$$

Simultaneously update
$w_j\ (for\ j = 1, \dots, n)\ and\ b$

An Alternative to gradient descent

- **Normal equation (alternative)**
  - Only for linear regression
  - Solve for w, b without iterations

"Disadvantages"

- Doesn't generalize to other learning algorithms. (like logistic regression)
- Slow when number of features is large (> 10,000)

"What you need to know"

- Normal equation method may be used in different ML library
- Gradient descent is the recommended method for finding w, b