

## 36 – Notation

## 36.1 - Notation

### Terminology

Training set: Data used to train the model

set:  $x$  size in feet?  $y$  price in \$1000's

(1)	2104	(400)
(2)	1416	232
(3)	1534	315
(4)	852	178
...	...	...
(47)	3210	870

$m = 47$

Notation:

$x$  = "input" variable

feature

$y$  = "output" variable

"target" variable

$m$  = number of training examples

$(x^{(i)}, y^{(i)})$  = single training example

$(x^{(i)}, y^{(i)})$

$(x^{(i)}, y^{(i)})$  =  $i^{\text{th}}$  training example

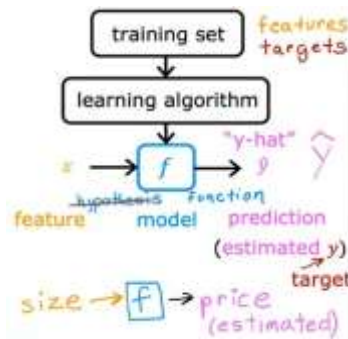
(1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> ...)

index

$$x^{(1)} = 2104 \quad y^{(1)} = 400$$

$$(x^{(1)}, y^{(1)}) = (2104, 400)$$

$$x^{(2)} = 1416 \quad x^{(2)} \neq x^2 \text{ not exponent}$$



How to represent  $f$ ?

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

$$f(x)$$



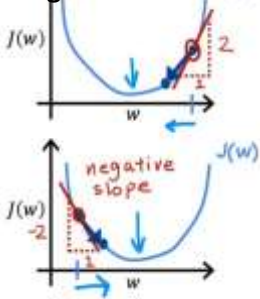
Linear regression with one variable.

Univariate linear regression.

one variable

## 101 - Gradient descent intuition

### Regression



$$w = w - \alpha \frac{d}{dw} J(w)$$

$$w = w - \alpha \cdot (\text{positive number})$$

$$\frac{d}{dw} J(w) > 0$$

$$\frac{d}{dw} J(w) < 0$$

$$w = w - \alpha \cdot (\text{negative number})$$

## 103 - Multiple Linear

### Multiple features (variables)

Size in feet <sup>2</sup>	Number of bedrooms	Number of floors	Age of home in years	Price (\$) in \$1000's
$x_1$	$x_2$	$x_3$	$x_4$	
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$x_j = j^{\text{th}}$  feature

$n$  = number of features

$x^{(i)}$  = features of  $i^{\text{th}}$  training example

$x_j^{(i)}$  = value of feature  $j$  in  $i^{\text{th}}$  training example

$$x^{(2)} = [1416 \ 3 \ 2 \ 40]$$

$$x_3^{(2)} = 2$$

## 104 - Vector

## 104 – Vector notation cost func

Parameters and features

$\vec{w} = [w_1 \ w_2 \ w_3]$   $n=3$

$b$  is a number

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

linear algebra: count from 1

code: count from 0

Without vectorization

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

$\vec{f} = w[0] * \vec{x}[0] +$

$w[1] * \vec{x}[1] +$

$w[2] * \vec{x}[2] + b$

Without vectorization

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

$\text{range}(0, n) \rightarrow j = 0 \dots n-1$

$\vec{f} = 0$

for  $j$  in  $\text{range}(0, n)$ :

$\vec{f} = \vec{f} + w[j] * \vec{x}[j]$

$\vec{f} = \vec{f} + b$

Vectorization

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

$\vec{f} = \text{np.dot}(\vec{w}, \vec{x}) + b$

$\vec{f} = \text{np.dot}(\vec{w}, \vec{x}) + b$

Previous notation

Parameters  $w_1, \dots, w_n$

$b$

Model  $f_{w,b}(\vec{x}) = w_1x_1 + \dots + w_nx_n + b$

Cost function  $J(w_1, \dots, w_n, b)$

Gradient descent

repeat {

$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots, w_n, b)$

$b = b - \alpha \frac{\partial}{\partial b} J(w_1, \dots, w_n, b)$

}

Vector notation

$\vec{w}$  = vector of length  $n$

$\vec{w} = [w_1 \ \dots \ w_n]$

$b$  still a number

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

$J(\vec{w}, b)$  dot product

repeat {

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

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

}

$$w = w - \alpha \frac{d}{dw} J(w)$$

Page : 101

If  $\alpha$  is too small...

Gradient descent may be slow.

By taking the smaller value of Learning Rate we will be taking smaller baby steps towards the least cost function. The steps are small hence the process is slow

If  $\alpha$  is too large...

Gradient descent may:

- Overshoot, never reach minimum
- Fail to converge, diverge

By taking the larger value for Learning Rate we will be taking the larger steps towards the least cost function that causes overshoot of values and resists reaching the minimum and fails to converge the values.

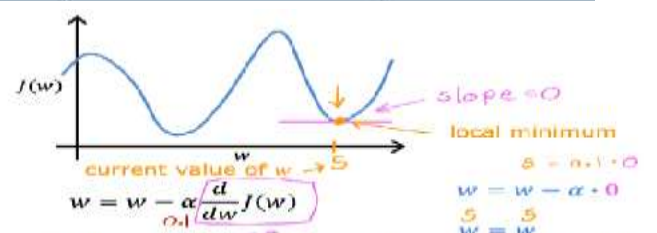
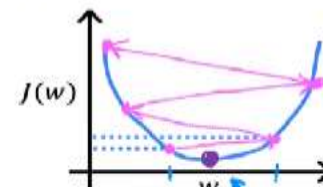
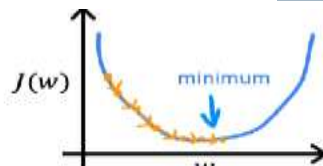
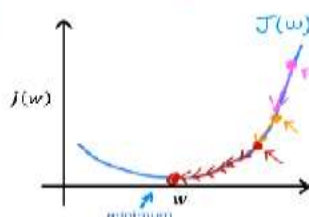
Can reach local minimum with fixed learning rate

$$w = w - \alpha \frac{d}{dw} J(w)$$

Near a local minimum,

- Derivative becomes smaller
- Update steps become smaller

Can reach minimum without decreasing learning rate  $\alpha$



Q. what if we reach the minimum cost function? will the gradient descent algo. update the value of  $w$  again?

A. Yes, the algorithm will change/update the value of  $w$  again n again but if the function has already reached the minimum point then it means that the slope (the derivative part of the formula) will be equal to 0 hence the value of  $w$  wont change and be always at local minima.

We can reach local minima while fixing the value of Learning Rate.

As we get closer to local minima the derivative (slope) part will decrease hence the step we take will be baby step.

## 104 - Vector Notation Gradient Descent(2)

### Gradient descent

One feature

repeat {

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

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

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

simultaneously update  $\underline{w}, b$

}

$n$  features ( $n \geq 2$ )

repeat {

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

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

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

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

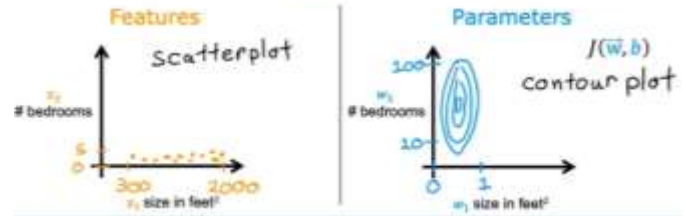
simultaneously update  $\underline{w}_j$  (for  $j = 1, \dots, n$ ) and  $b$

}

## 105 - Feature Scaling (1)

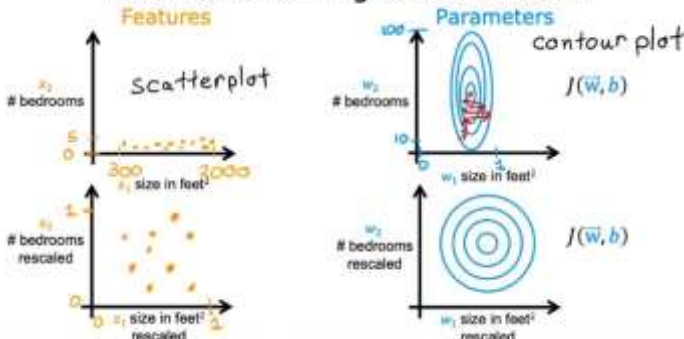
### Feature size and parameter size

	size of feature $x_j$	size of parameter $w_j$
size in feet <sup>2</sup>	→	→
#bedrooms	→	→



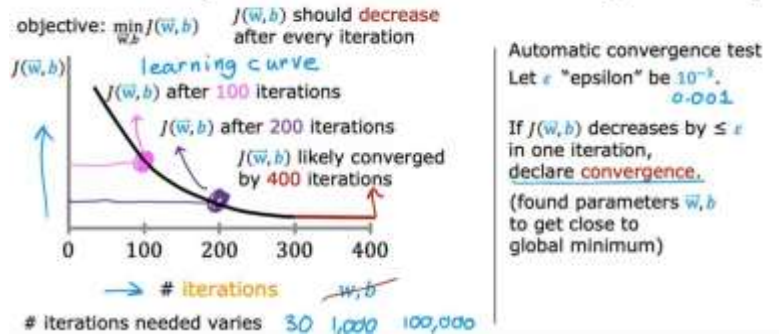
## 105 - Feature Scaling (2)

### Feature size and gradient descent



## 105 - Feature Scaling (3)

### Make sure gradient descent is working correctly

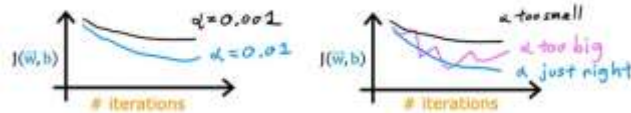


## 105 - Learning Rate (1)

Values of  $\alpha$  to try:

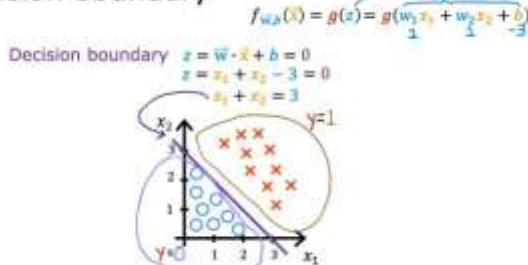
... 0.001 0.003 0.01 0.03 0.1 0.3 1 ...

$3x \approx 3x$   $3x \approx 3x$   $3x \approx 3x$   $3x \approx 3x$



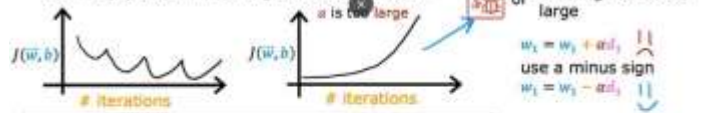
## 110 - Decision Boundary (1)

### Decision boundary



## 105 - Learning Rate

### Identify problem with gradient descent

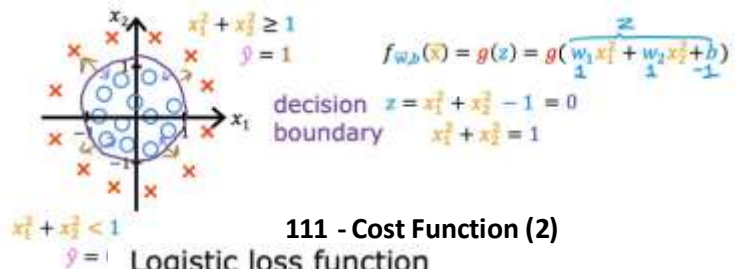


### Adjust learning rate



## 110 - Decision Boundary (2)

### Non-linear decision boundaries



## 111 - Cost Function (1)

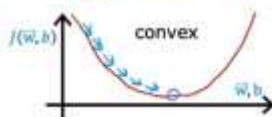
### Squared error cost

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

log loss  $L(f_{\underline{w},b}(\underline{x}^{(i)}), y^{(i)})$

linear regression

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



logistic regression

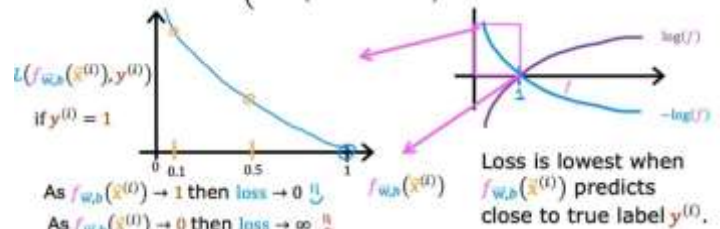
$$f_{\underline{w},b}(\underline{x}) = \frac{1}{1 + e^{-(\underline{w} \cdot \underline{x} + b)}}$$



## 111 - Cost Function (2)

### Logistic loss function

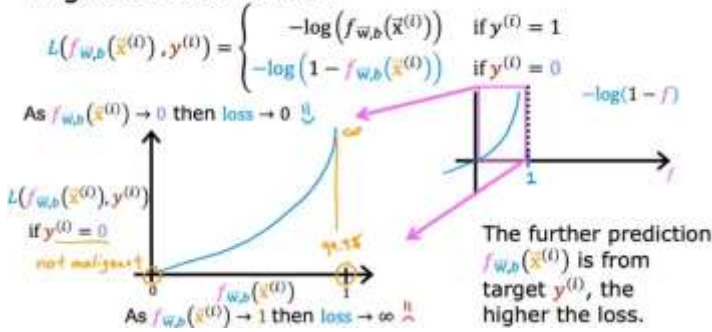
$$L(f_{\underline{w},b}(\underline{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(f_{\underline{w},b}(\underline{x}^{(i)})) & \text{if } y^{(i)} = 1 \\ -\log(1 - f_{\underline{w},b}(\underline{x}^{(i)})) & \text{if } y^{(i)} = 0 \end{cases}$$





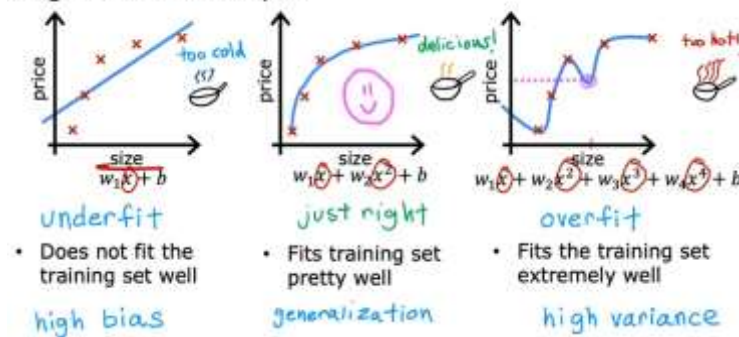
## 111 - Cost Function (3)

### Logistic loss function



## Overfitting & Underfitting (1)

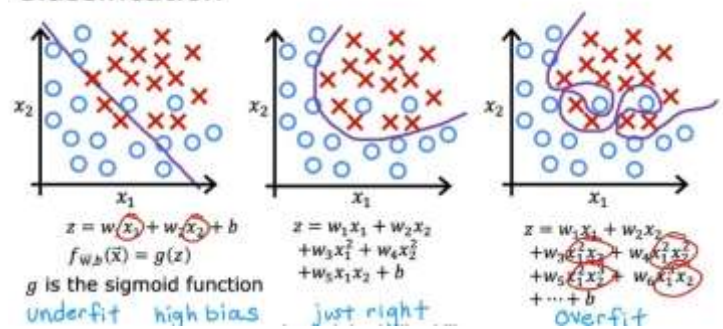
### Regression example



## Overfitting & Underfitting (3)

## Overfitting & Underfitting (2)

### Classification



## 115 - Regularization for Logistics Regression

### Regularized logistic regression

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

### Gradient descent

repeat {

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

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

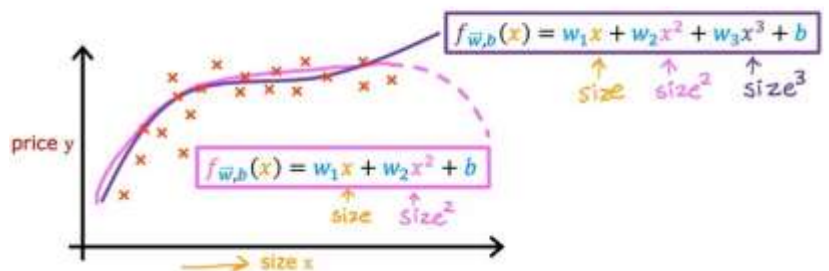
}

Looks same as for linear regression!

logistic regression

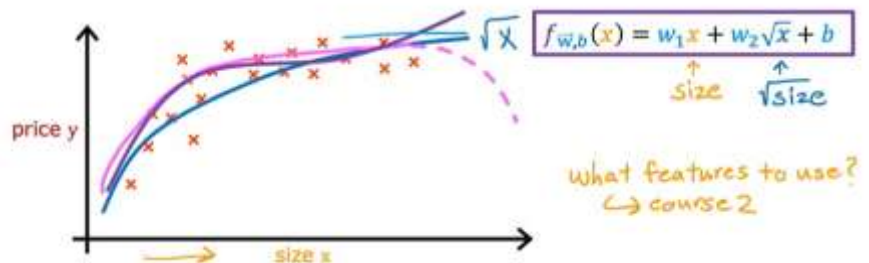
## Polynomial Regression (1)

### Polynomial regression



## Polynomial Regression (2)

### Choice of features



### Overfitting: Key definitions

Here are some of the key definitions that'll help you navigate through this guide.

- Bias:** Bias measures the difference between the model's prediction and the target value. If the model is oversimplified, then the predicted value would be far from the ground truth resulting in more bias.
- Variance:** Variance is the measure of the inconsistency of different predictions over varied datasets. If the model's performance is **tested on different datasets**, the closer the prediction, the lesser the variance. Higher variance is an indication of overfitting in which the model loses the ability to generalize.
- Bias-variance tradeoff:** A simple linear model is expected to have a high bias and low variance due to less complexity of the model and fewer trainable parameters. On the other hand, complex non-linear models tend to observe an opposite behavior. In an ideal scenario, the model would have an optimal balance of bias and variance.
- Model generalization:** Model generalization means how well the model is trained to extract useful data patterns and classify unseen data samples.
- Feature selection:** It involves selecting a subset of features from all the extracted features that contribute most towards the model performance. Including all the features unnecessarily increases the model complexity and redundant features can significantly increase the training time.