ML SPECIALIZATION COURSERA

1. Supervised vs Unsupervised ML

a) Supervised learning

- In supervised learning, the model is trained on input-output pairs, where the goal is to learn the mapping from input X to output Y

Example:

email -> spam (0/1) - binary classification
audio -> text transcipt - speech-to-text transcript
English -> Spanish - machine translation
input -> housing prices - regression

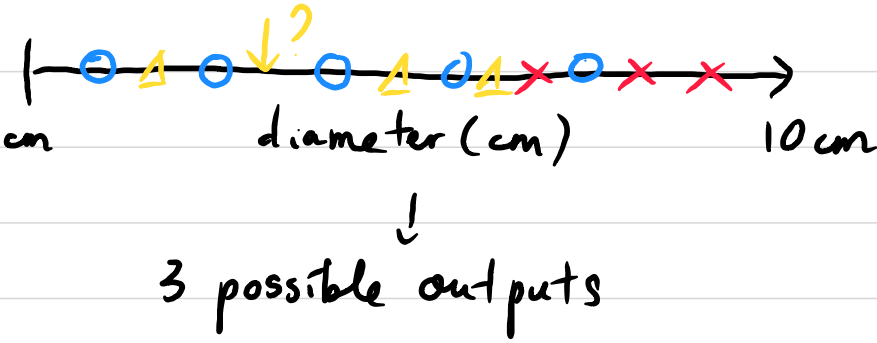
Classification: Breast cancer detection
(malignant or benign) - (ác tính or lành tính)

size	diagnosis
2	0
5	1
1	0
7	1

O benign

X malignant type 1

Δ malignant type 2

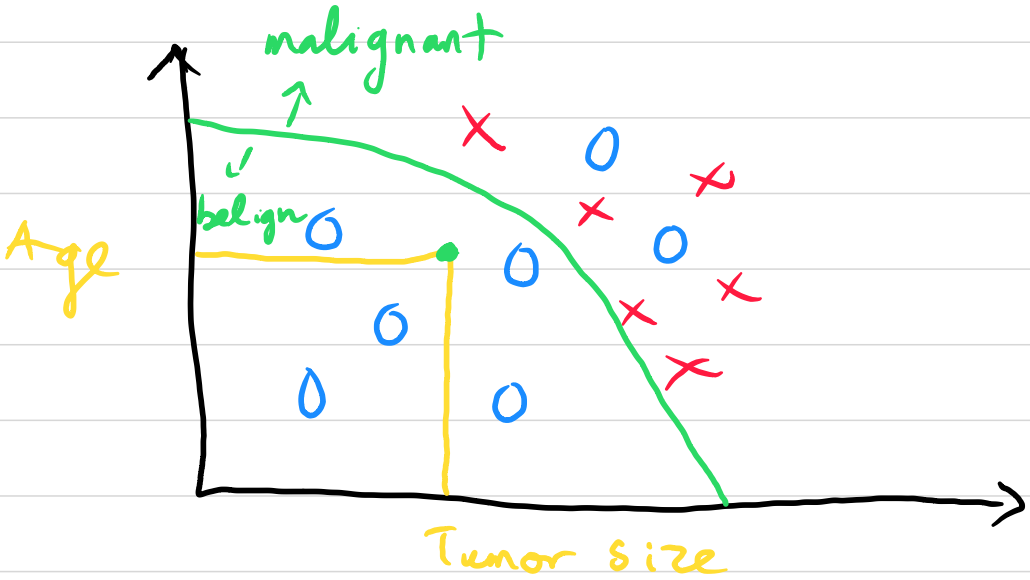


3 possible outputs

Classification

predict categories 0, 1, 2

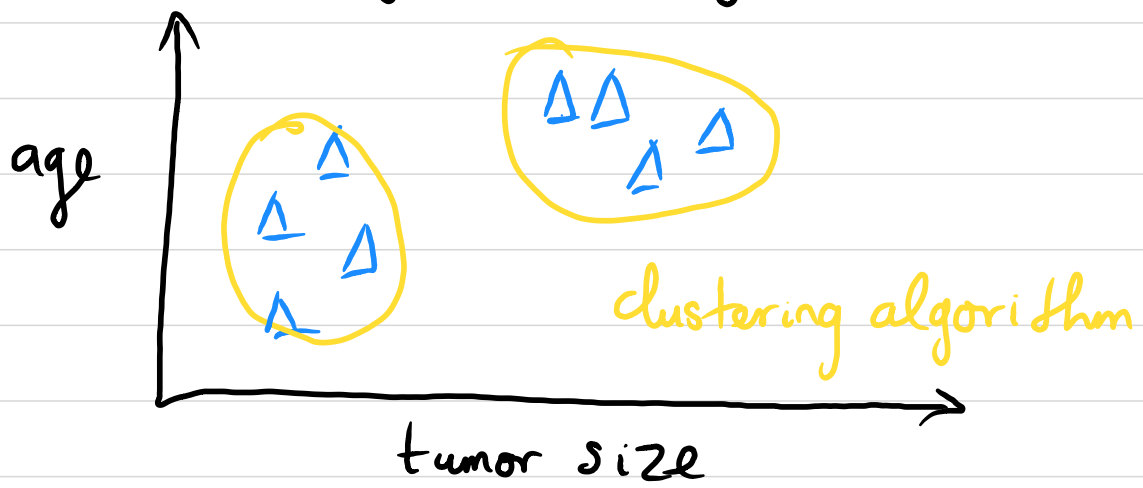
Two or more inputs



- the algorithm has to decide how to fit a boundary line
- Supervised learning learns from being given "right answers"

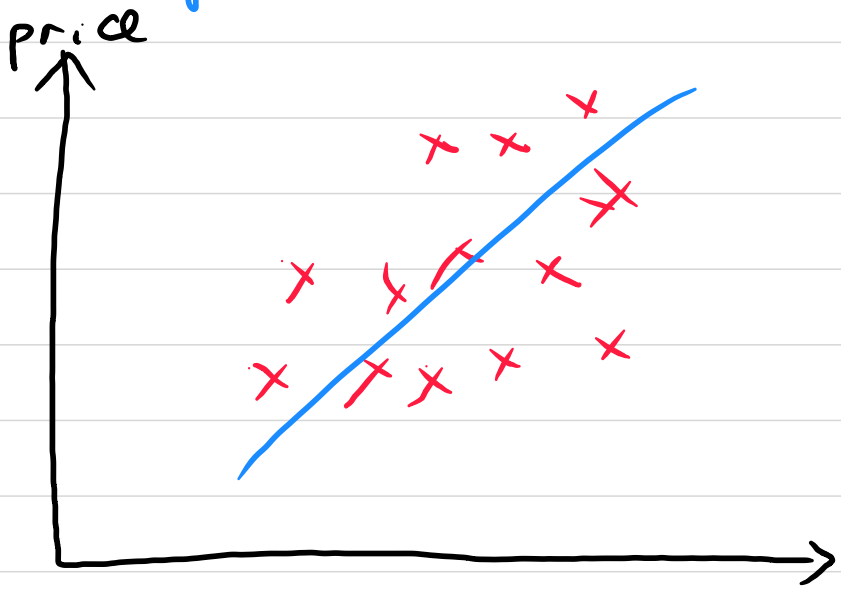
b) Unsupervised learning

- Find something interesting in "unlabeled" data



- Clustering: group similar data points together
- Anomaly detection: find unusual data points
- Dimensionality reduction: compress data using fewer numbers

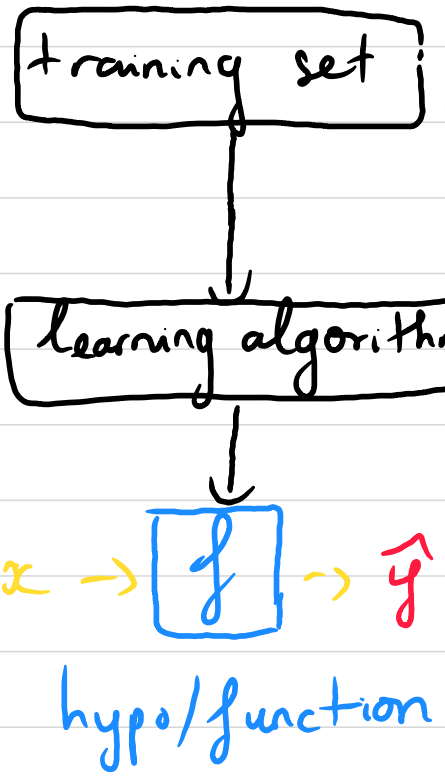
2. Regression model



Data table (train)

size (x)	price (y)
2104	400
1406	232
1534	315
852	178
...	...
3210	870

- x = input/feature variable
- y = output/target variable
- m = number of training examples
- (x, y) = single training example
- $(x^{(i)}, y^{(i)})$ = i^{th} training example

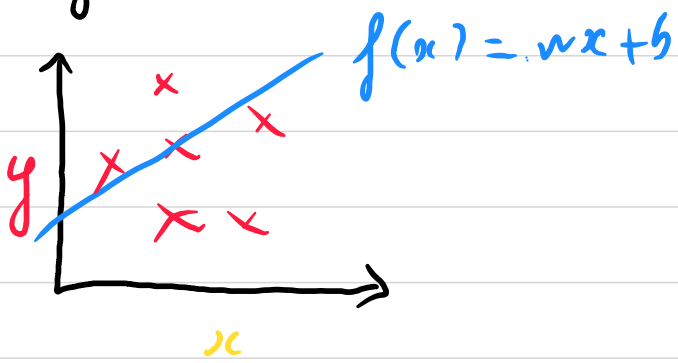


feature → model → prediction (estimated y)

How to represent f ?

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

$f(x)$



Cost function

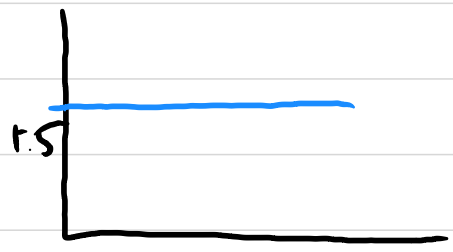
Train set

x	y
2104	460
1153	232
...	...

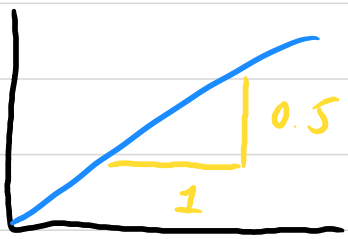
model $f_{w,b}(x) = wx + b$

w, b : parameters
coefficients
weights

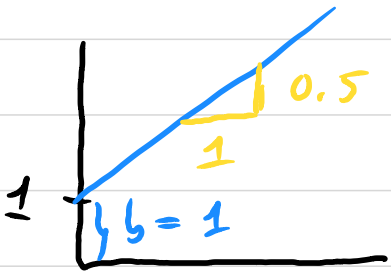
$f(x) = wx + b$



$w = 0$
$b = 1.5$
intercept



$w = 0.5$
$b = 0$
slope



$w = 0.5$
$b = 1$

Target: Find w, b:

$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$

⇒ Construct the cost function

(square error cost function)

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \underbrace{\left(\hat{y}^{(i)} - y^{(i)} \right)^2}_{\text{error}}$$

m: # training examples

commonly used for linear regression

model:

$$f(x) = wx + b$$

parameters:

w, b

cost function:

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

goal:

minimize $J(w,b)$

w, b

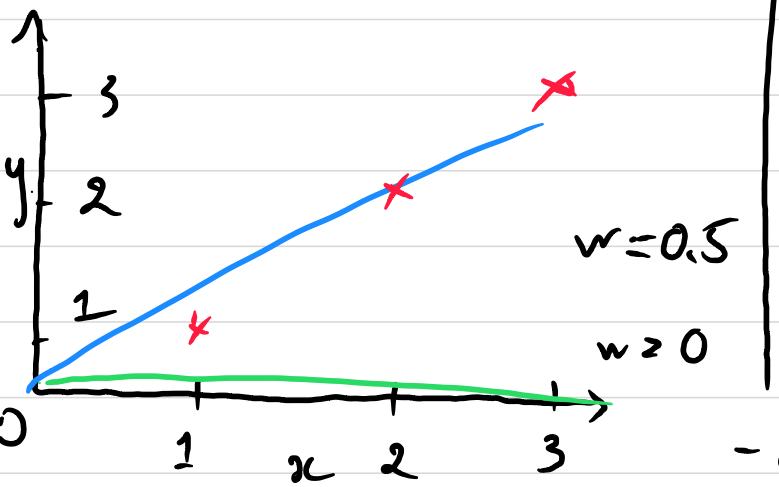
simplified: $b = 0$

$$f(x) = wx$$

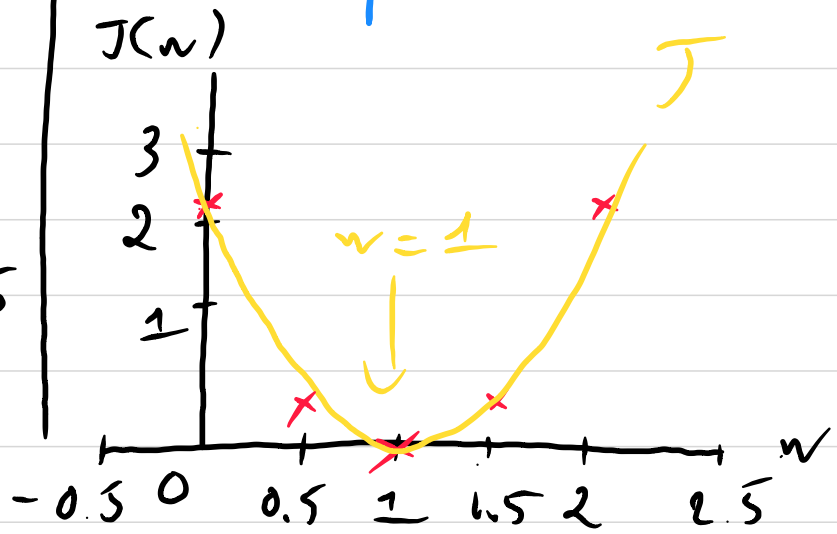
⇒ $J(w,b) = J(w) = \frac{1}{2m} \left(f_w(x^{(i)}) - y^{(i)} \right)^2$

⇒ minimize $J(w)$

$f_w(x)$
(for fixed w, function of x)
input



$J(w)$
(function of w)
parameter



$$J(0.5) = \frac{1}{2m}\left[(1.5-1)^2 + (2-2)^2 + (2.5-3)^2\right]$$

$$= \frac{1}{12}$$

$$J(0) = \frac{1}{2m}\left[(0-1)^2 + (0-2)^2 + (0-3)^2\right]$$

$$= 7/3$$

- It's hard trying to find w, b manually by subtituing values to $f(x)$ and J

=> GRADIENT DESCENT

3. Gradient descent

Have some function $J(w, b)$

Want:

$$\min J(w, b)$$

How?

1. Start with initial w, b (set $w = b = 0$)
2. Keep changing w, b to reduce J until we settle at or near a minimum

↓

may have ≥ 1 minimum

Gradient descent algorithm

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

→ where to step

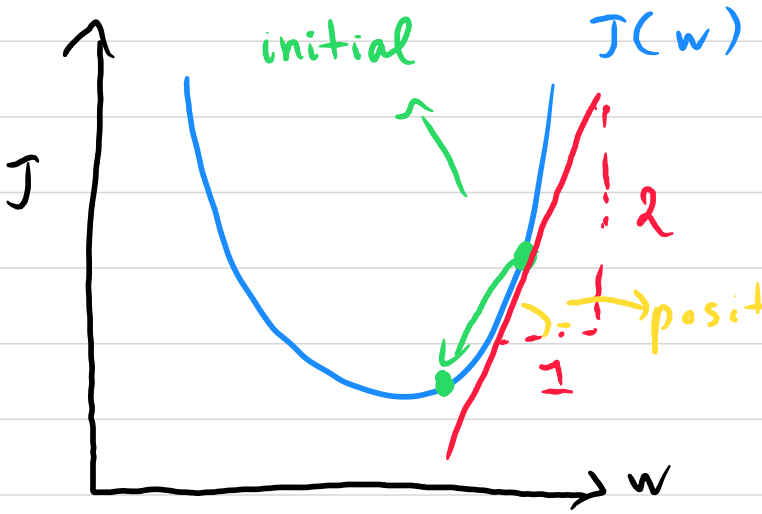
α: learning rate (step)

$$b = b - \alpha \cdot \frac{d}{db} J(w, b)$$

Correct approach: Simultaneously update

1. $temp\_w = w - \alpha \cdot \frac{d}{dw} J(w, b)$
2. $temp\_b = w - \alpha \frac{d}{db} J(w, b)$
3. $w = temp\_w$
4. $b = temp\_b$

Gradient descent intuition

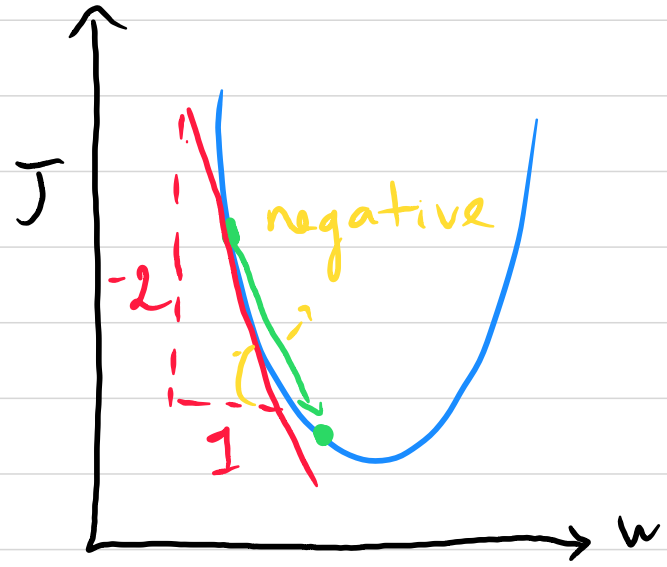


> 0

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

positive slope

$\Rightarrow w = w - \alpha$ positive num



$w = w - \alpha$. negative num

Learning rate

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

- If α <<, small steps, gradient descent slow
- If α >>, large steps, cost maybe got worse
- If already in local minimam, slope = 0,

$w = w - \alpha . 0 \Rightarrow w$ unchanged

Gradient descent for Linear Regression

LR model

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

Cost function

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

Gradient descent

repeat until convergence{

$$w = w - \alpha \frac{d}{dw} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f(x^{(i)}) - y^{(i)}\right)x^{(i)}$$

$$b = b - \alpha \frac{d}{db} J(w,b) \rightarrow \frac{1}{m}\sum_{i=1}^{m}\left(f(x^{(i)}) - y^{(i)}\right)$$

Transformation (optional)

$$\frac{d}{dw} J(w,b) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right).2.x^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)x^{(i)}$$

$$\frac{d}{db} J(w,b) = \frac{d}{db}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right).2$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)$$

4. LR with multiple features

x_1	x_2	x_3	x_4	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

n = # features

$\vec{x}^{(i)}$ = features of i^{th} training example (row vector)

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

Model: (4 features)

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + w_3 x_4 + w_4 x_4 + b$$

General: (n features)

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

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

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

b is a constant

⇒ Rewriten form: → dot product

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

5 Vectorization

Params and features

$\vec{w} = [w_1, w_2, w_3]$; $\vec{x} = [x_1, x_2, x_3]$; b

Without vectorization

1) $f = w[0] \cdot x[0] + w[1] \cdot x[1] + w[2] \cdot x[2] + b$

2) $f = 0$

```
for i in range(n):
    f = f + w[i] . x[i]
f = f + b
```

Vectorization

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

Gradient descent

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

repeat {

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

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

}

One feature

repeat {

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

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

simultaneously update w, n

}

n features ($n \geq 2$)

repeat {

$j = 1$

$$w_1 = w_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (f(\vec{x}^{(i)}) - y^{(i)}) x_1^{(i)}$$

. . . . . - - - -

w_n ($j = n$)

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

simultaneously update w ($j = 1, 2, \ldots n$) and b

}

6. Feature scaling

- When the possible values of the feature are small, the relative parameter will be large, vice versa

Mean normalization

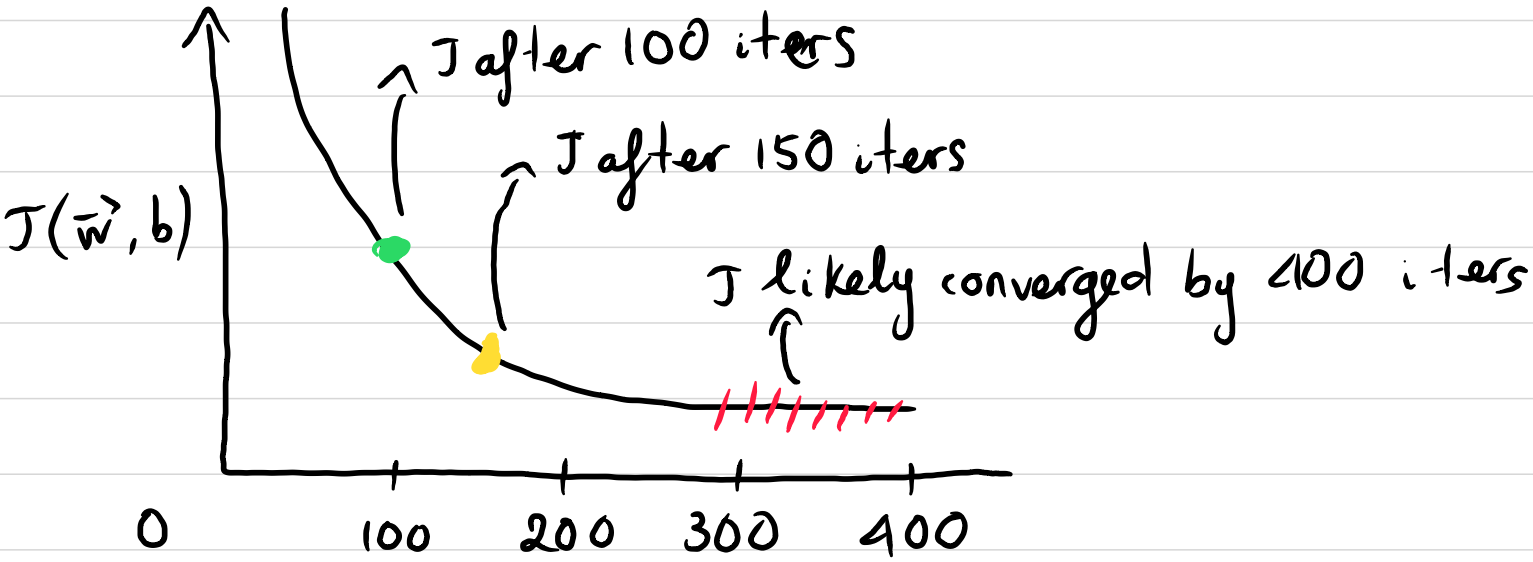
$$x = \frac{x - \mu}{Max - Min}$$

Z-score normalization

$$x = \frac{x - \mu}{\sigma}$$

Check gradient descent for Convergence

Graphical approach (more feasible)



Automatic converge test

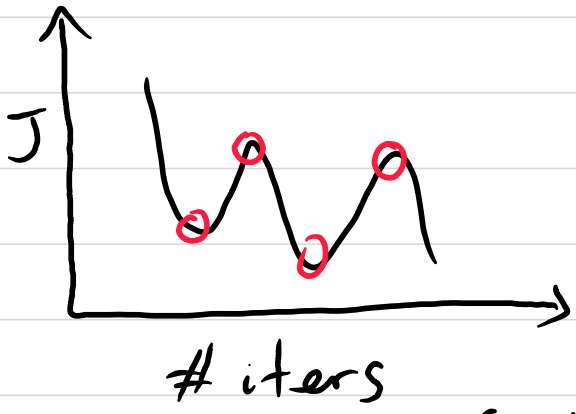
Let ϵ be 10^{-3}

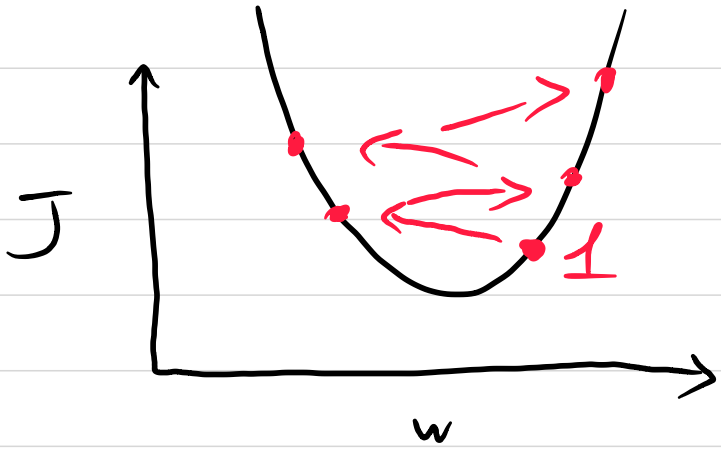
If J decrease by $\Delta \leq \epsilon$ in 1 iter,

declare convergence

Note: hard to find suitable ϵ

Choose learning rate





Cost goes up and down

⇒ Bug or Poor α

- With a small enough α, $J(\vec{w}, b)$ should decrease on every iteration

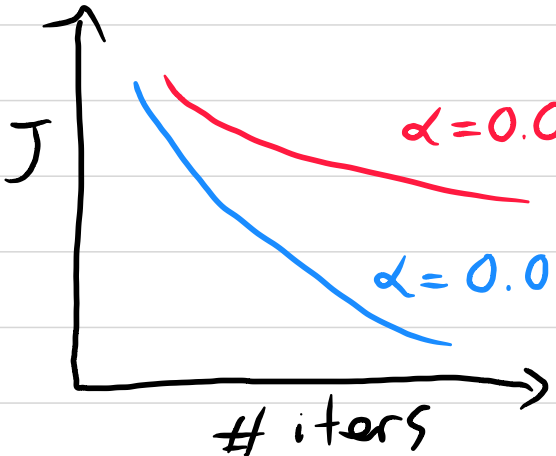
- If $\alpha \ll$, gradient descent takes a lot more iters to converge

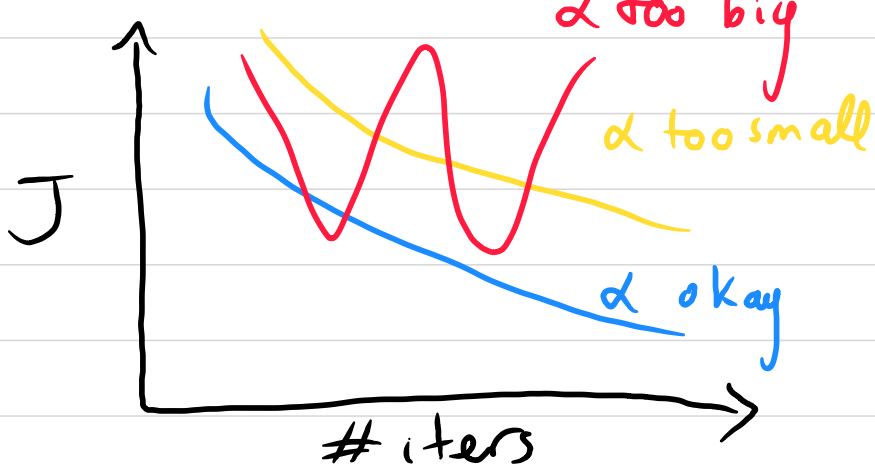
Approach for good α

0.001 → 0.003 → 0.01 → 0.03 → 0.1 → 0.3 → 1

3x 3x 3x 3x 3x 3x

- Plot out the J for each chosen α



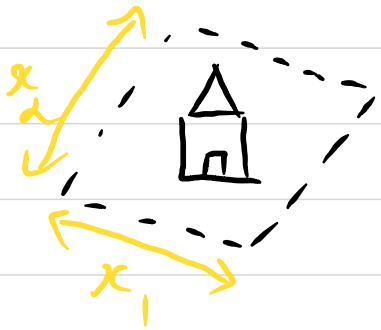


7. Feature Engineering

- Design new features, by transforming and combining original features

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

x_1: frontage, x_2: depth



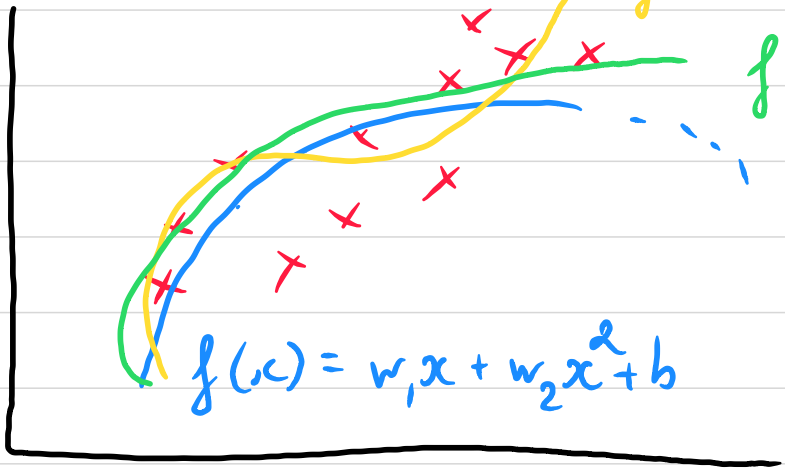
area = frontage × depth

$\Leftrightarrow$ $x_3 = x_1 x_2$

→ new feature : area

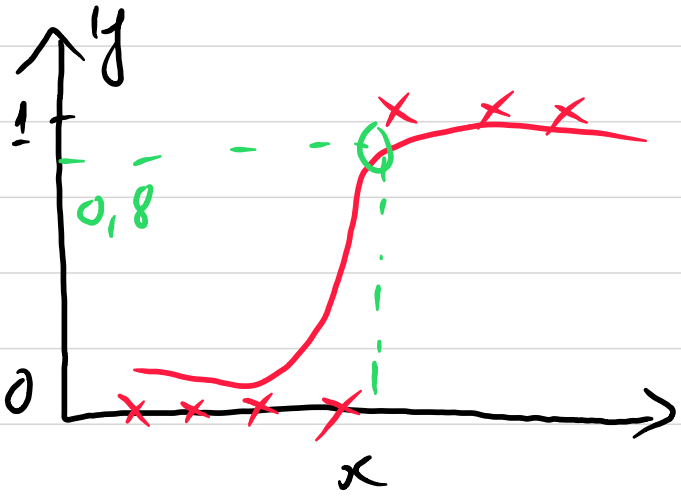
$$f_{\vec{w},b}(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

8. Polynomial regression

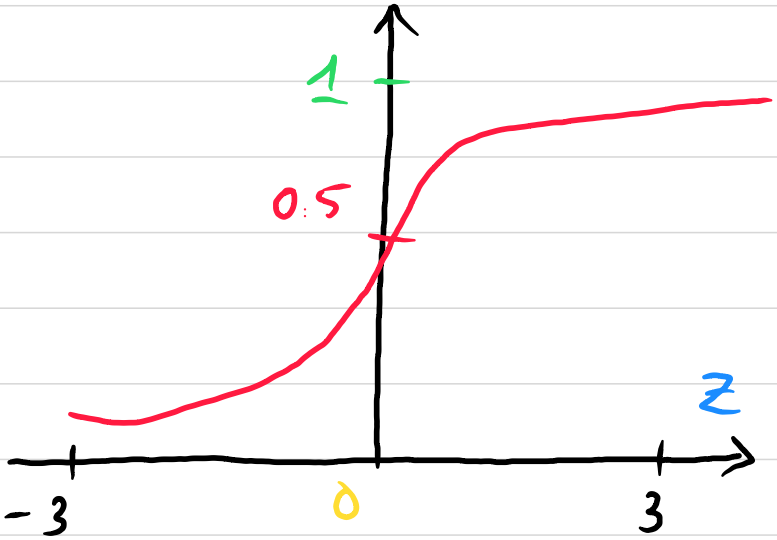


9 Classification

Logistic regression



Want outputs between 0 and 1



sigmoid function

or logistic function

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

$z = \vec{w} \cdot \vec{x} + b$

z

$g(z) = \frac{1}{1+e^{-z}}$

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

"logistic regression"

outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}} \quad 0 < g(z) < 1$$

Output of logistic regression

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

$\Rightarrow f_{\vec{w},b}(\vec{x})$ is the probability that class is 1

Example:

x : tumor size

y : 0 (not malignant)
1 (malignant)

We have $f_{\vec{w},b}(\vec{x}) = 0.7$ which means that there is 70% chance that y is 1

Notation: $f_{\vec{w},b}(\vec{x}) = P(y=1 \mid \vec{x}; \vec{w}, b)$

Decision boundary

Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

yes: $\hat{y} = 1$; no: $\hat{y} = 0$

When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(z) \geq 0.5$

$z \geq 0$

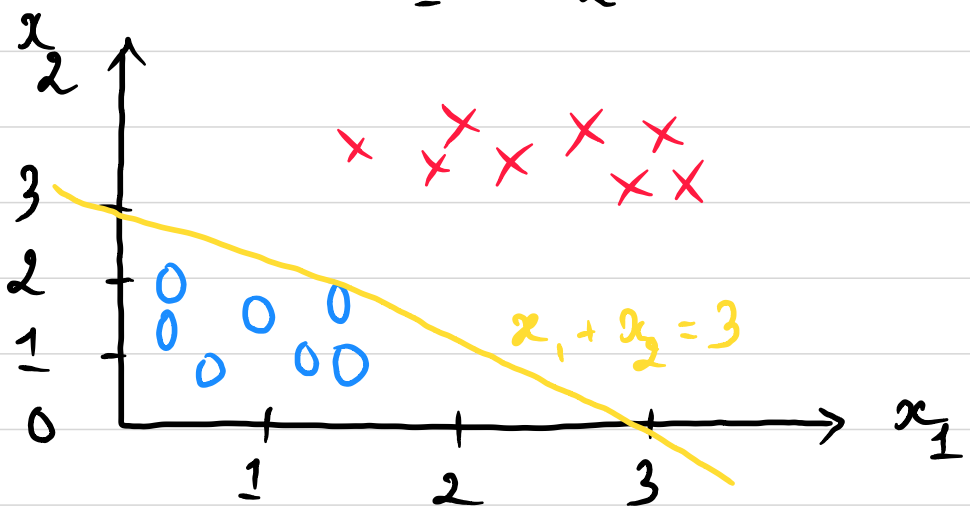
$\Rightarrow \vec{w}\cdot\vec{x} + b \geq 0$ | $\vec{w}\cdot\vec{x} + b < 0$

$\hat{y} = 1$ | $\hat{y} = 0$

$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + b)$$
$$1 \qquad 1 \qquad -3$$

Decision boundary $z = \vec{w} \cdot \vec{x} + b = 0$

$$= x_1 + x_2 - 3 = 0$$
$$x_1 + x_2 = 3$$

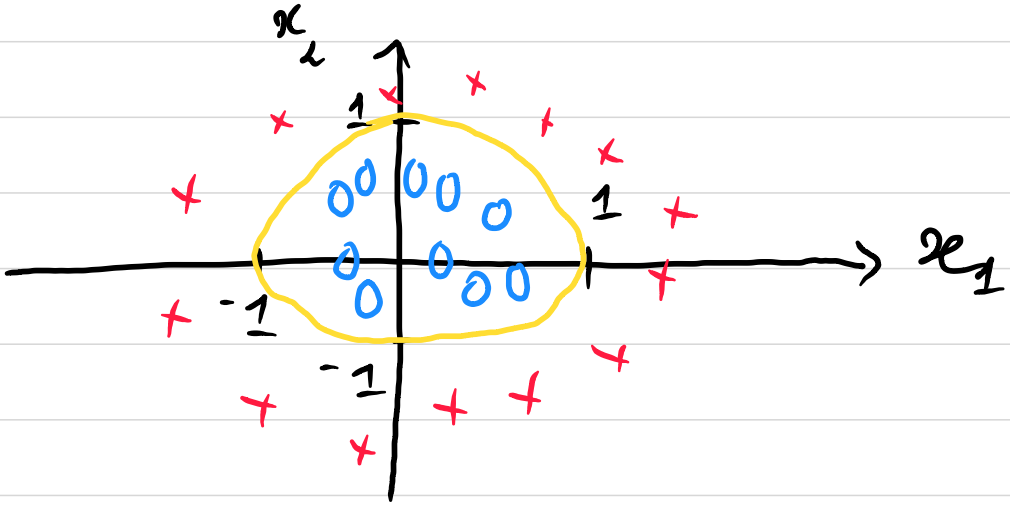


Non-linear decision boundaries

$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 x_1^2 + w_2 x_2^2 + b)$$
$$1 \qquad 1 \qquad -1$$

Decision boundary $z = x_1^2 + x_2^2 - 1 = 0$

$$x_1^2 + x_2^2 = 1$$



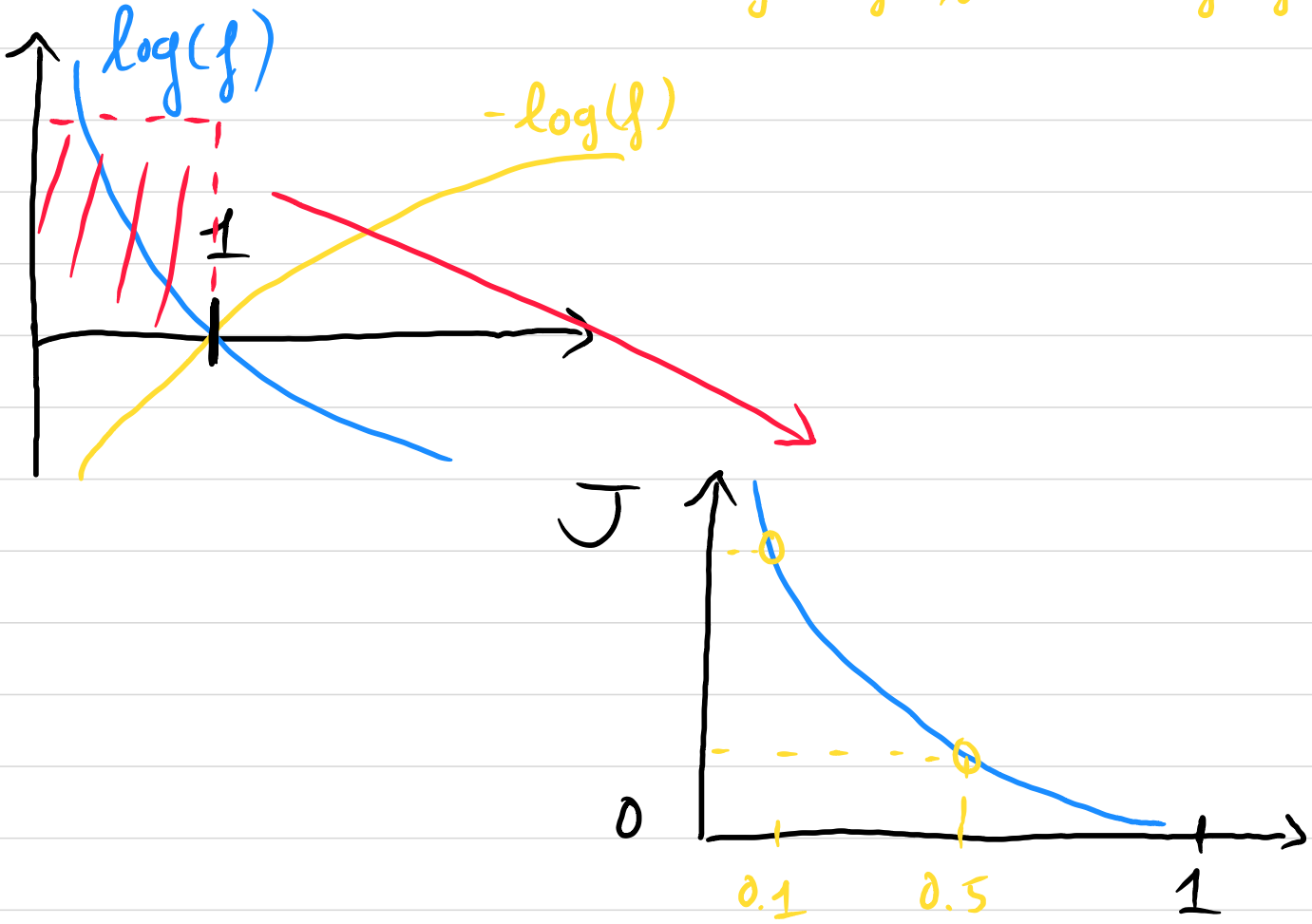
Cost function for logistic regression

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

$$\downarrow$$

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)} \right)$$

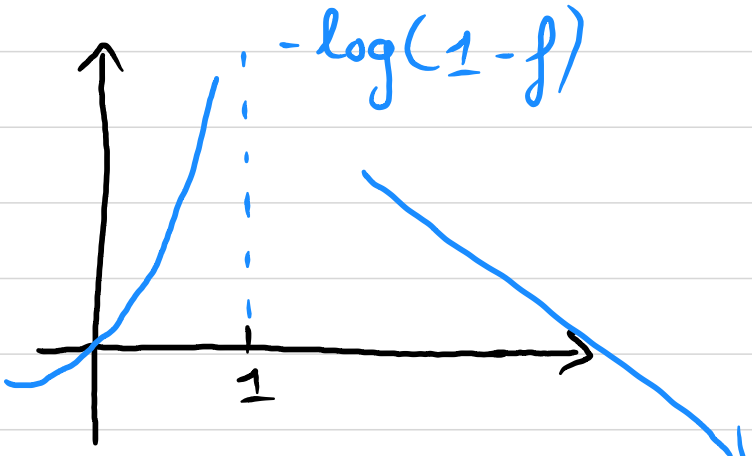
$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)} \right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)}) \right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)}) \right) & \text{if } y^{(i)} = 0 \end{cases}$$

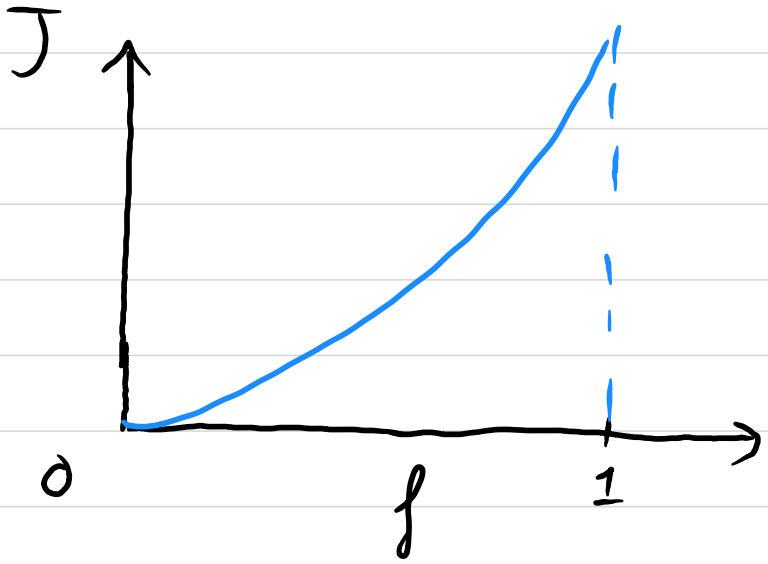


- Loss is lowest when $f_{\vec{w},b}(\vec{x}^{(i)})$ predicts close to true label $y^{(i)}$

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss $\to 0$

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to \infty$





As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$ then loss $\to 0$

As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$ then loss $\to \infty$

Simplified loss function

convex (single global min)

$$L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})$$

$$= -y^{(i)} \log(f_{\vec{w},b}(\vec{x}^{(i)})) - (1-y^{(i)}) \log(1 - f_{\vec{w},b}(\vec{x}^{(i)}))$$

$$J(\vec{w},b) = \frac{1}{m} \sum L(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)})$$

Logistic gradient descent

repeat {

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

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

} simultaneously update

where

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

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

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

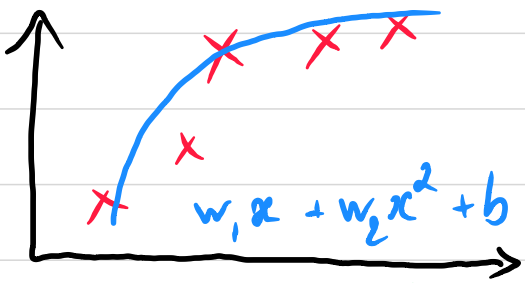
Logistic regression $f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$

10 Overfitting

underfit

does not fit the train set well

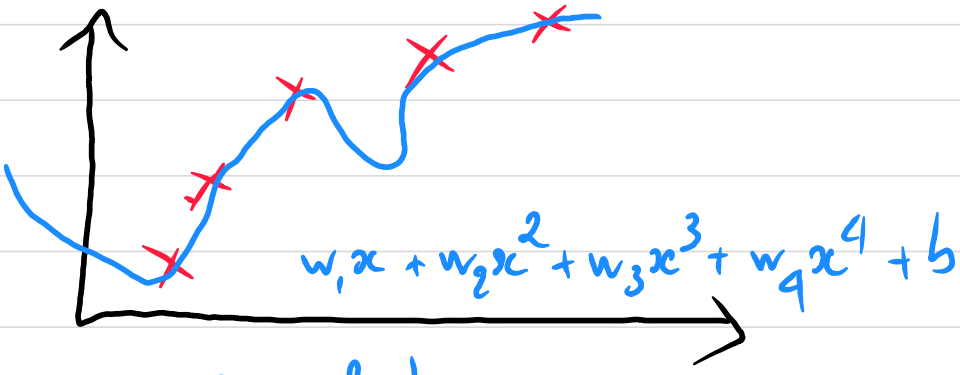
high bias



Just right

fits the train set pretty well

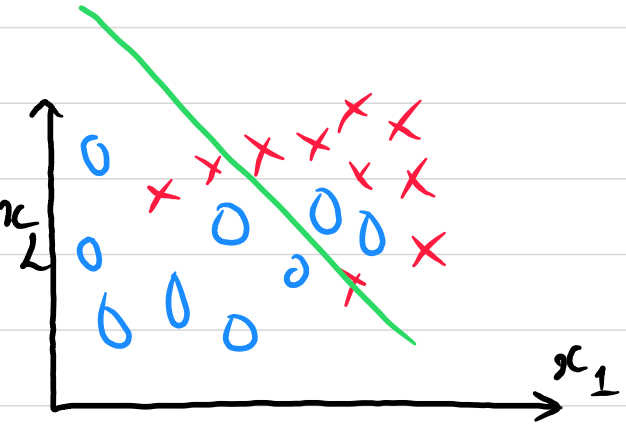
generalization

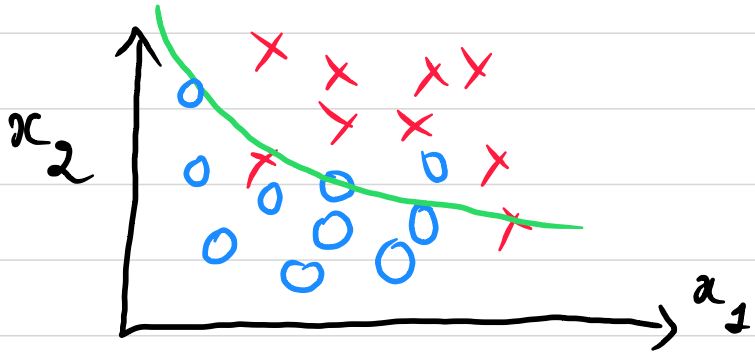


overfit

fits the train set extremely well

high variance





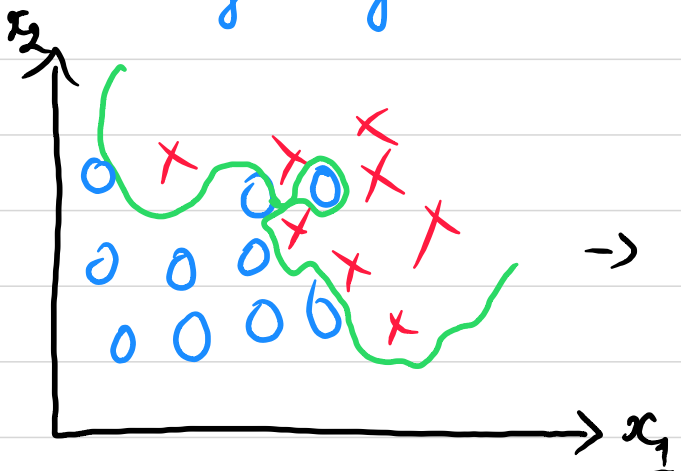
$$z = w_1x_1 + w_2x_2 + b$$

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

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

g is the sigmoid func

underfit/high bias



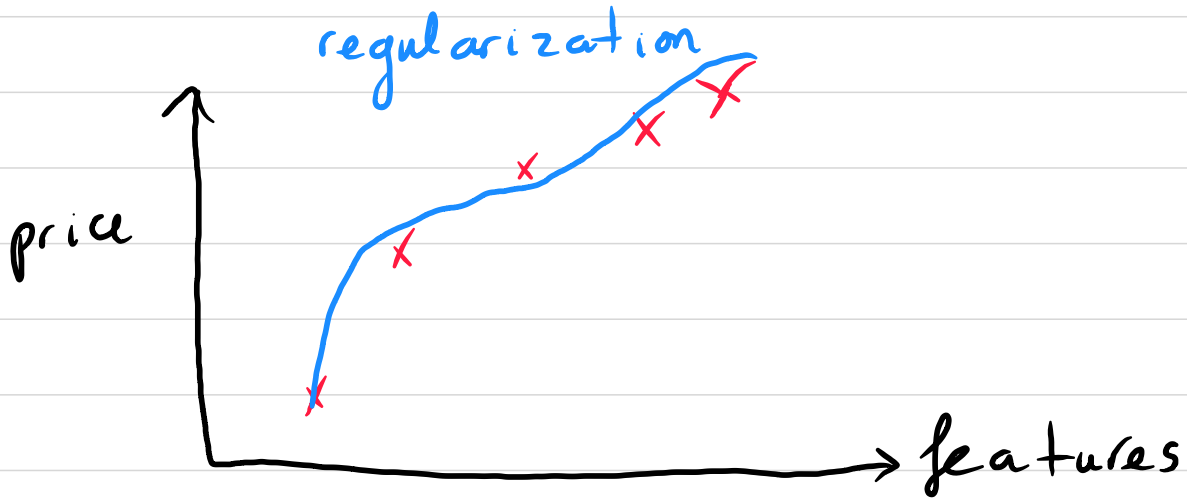
generalize

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

overfit

Address overfitting

- Collect more training examples
- Select features to include/exclude (feature selection)
- Regularization



$$f(x) = 13x - 0.23x^2 + 0.000014x^3 - 0.0001x^9 + 10$$

small values for w_j

- Regularization: keeps all features but prevent the features from having an overly large effect

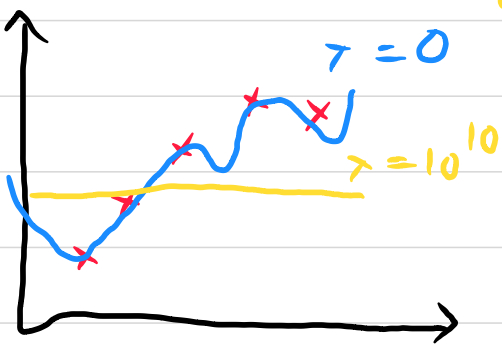
Cost function with regularization

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

MSE → fit data

regularization term → keep w_j small

→ balances both goals



Regularized linear regression

Gradient descent

repeat {

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

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

} simultaneously update

where:

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

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

↓ don't have to regularize b

Rewrite w_j:

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

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

shrink w_j — usual update

Derivative

$$\frac{d}{dw_j} J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^{m} \left[\left(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)} \right) 2 x_j^{(i)} \right] + \frac{\lambda}{2m} \cdot 2 w_j$$

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

Regularized logistic regression

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

Gradient descent

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

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