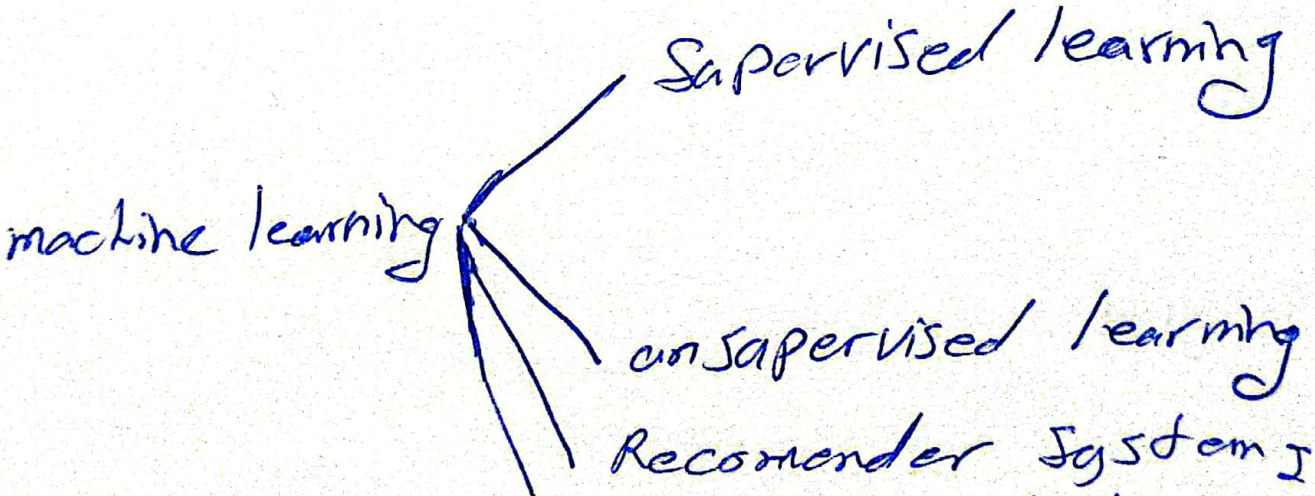
"supervised machine learning"

(regression and classification)

machine learning:

"field of study that gives computers the ability to learn without being explicitly programmed" Arthur Samuel (1959)



~~Supervised learning~~ 2

Supervised learning:

$x \longrightarrow y$

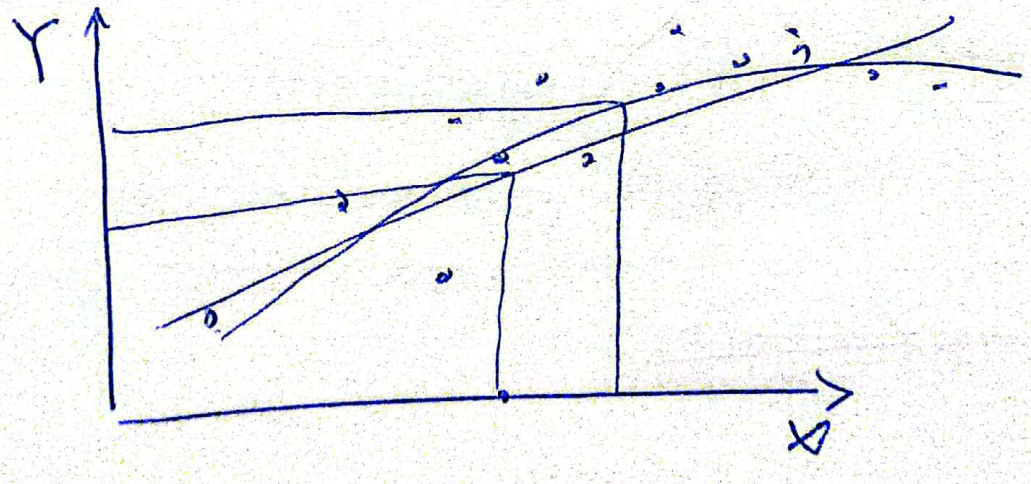
input ——> output label

learns from being given "right answers"

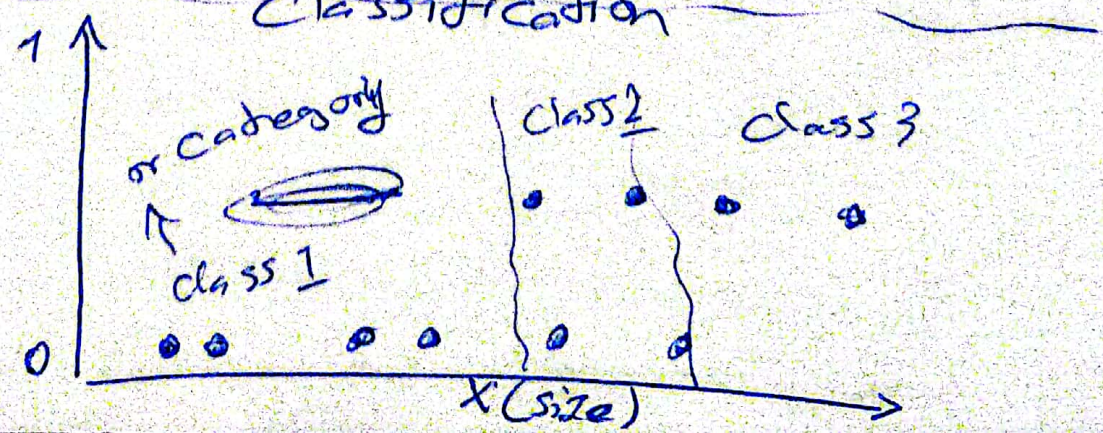
Supervised learning

input (X)	output (Y)	application
email	spam (0/1)	spam filtering
audio	text transcript	speech recognition
English	Spanish	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	self-driving car
image of phone	defect? (0/1)	visual inspection

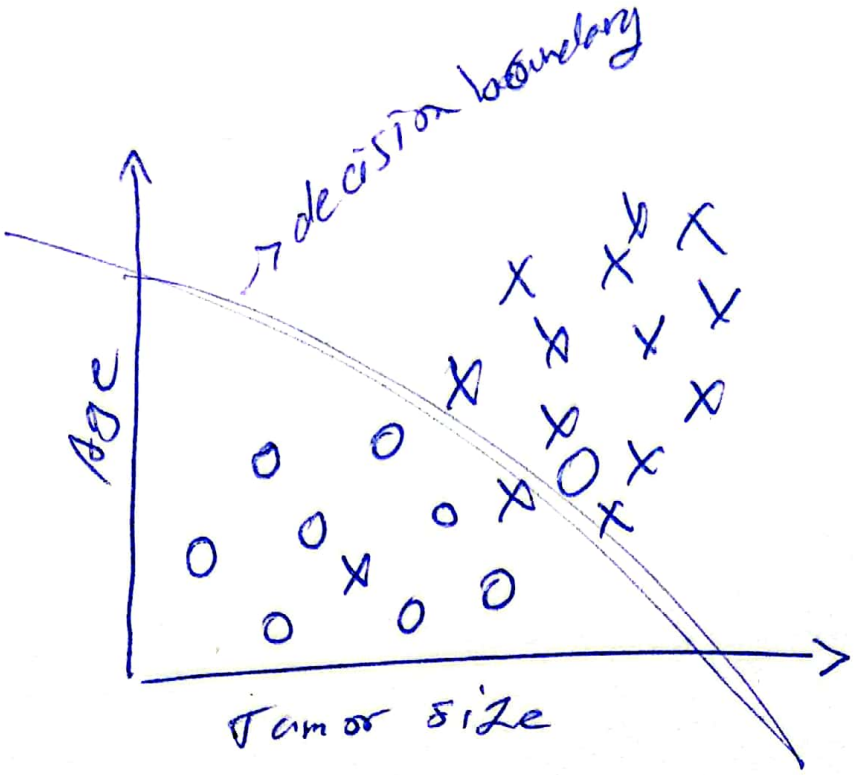
regression



classification



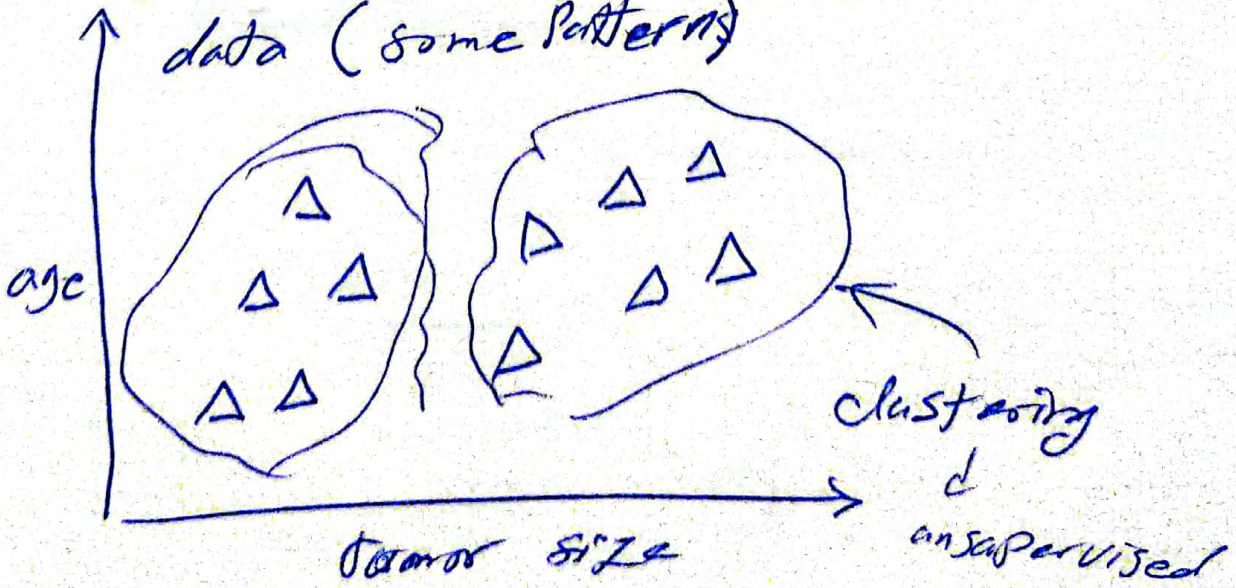
exp2: classification (2 Xs)



Supervised learning
- regression
- classification

unsupervised learning

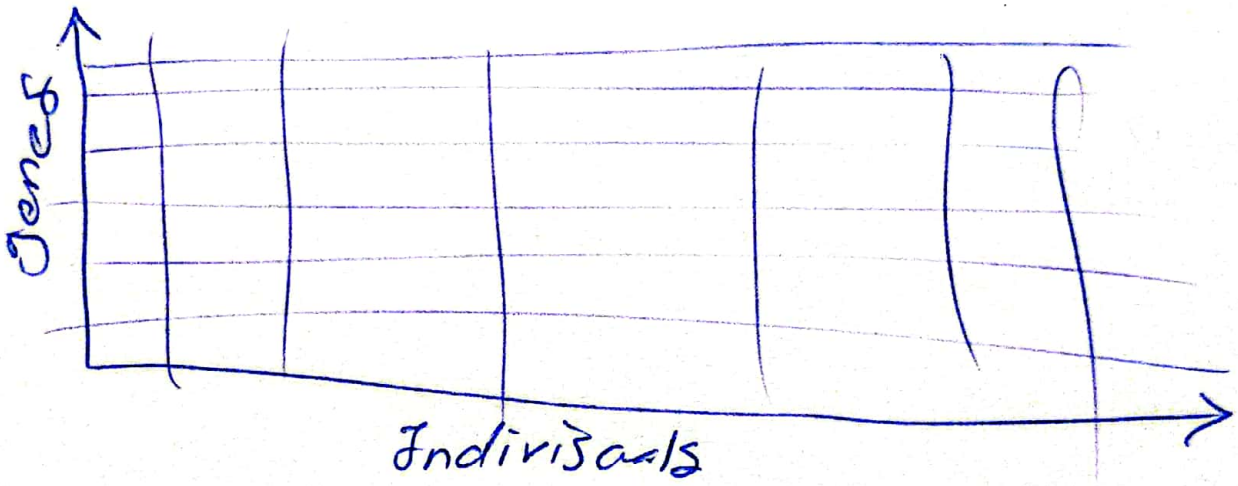
find something interesting in unlabled data (some Patterns)



Exps of unsupervised learning

(suggestion) Google news (clustering)

- clustering: DNA microarray



- clustering: Grouping customers

---

unsupervised learning:

Data only comes with inputs x, but not output labels y. algorithm has to find structure in the data.

Clustering

Group similar data points together

Anomaly detection

find unusual data points.

Dimensionality reduction

compress data using fewer numbers.

Linear regression model

Terminology

- Training set: Data used to train the model.
- x (lower case): input variable or feature
- y (lower case): output variable or target variable

(X) size in feet	(Y) price in $
2014	400
1416	232
⋮	⋮

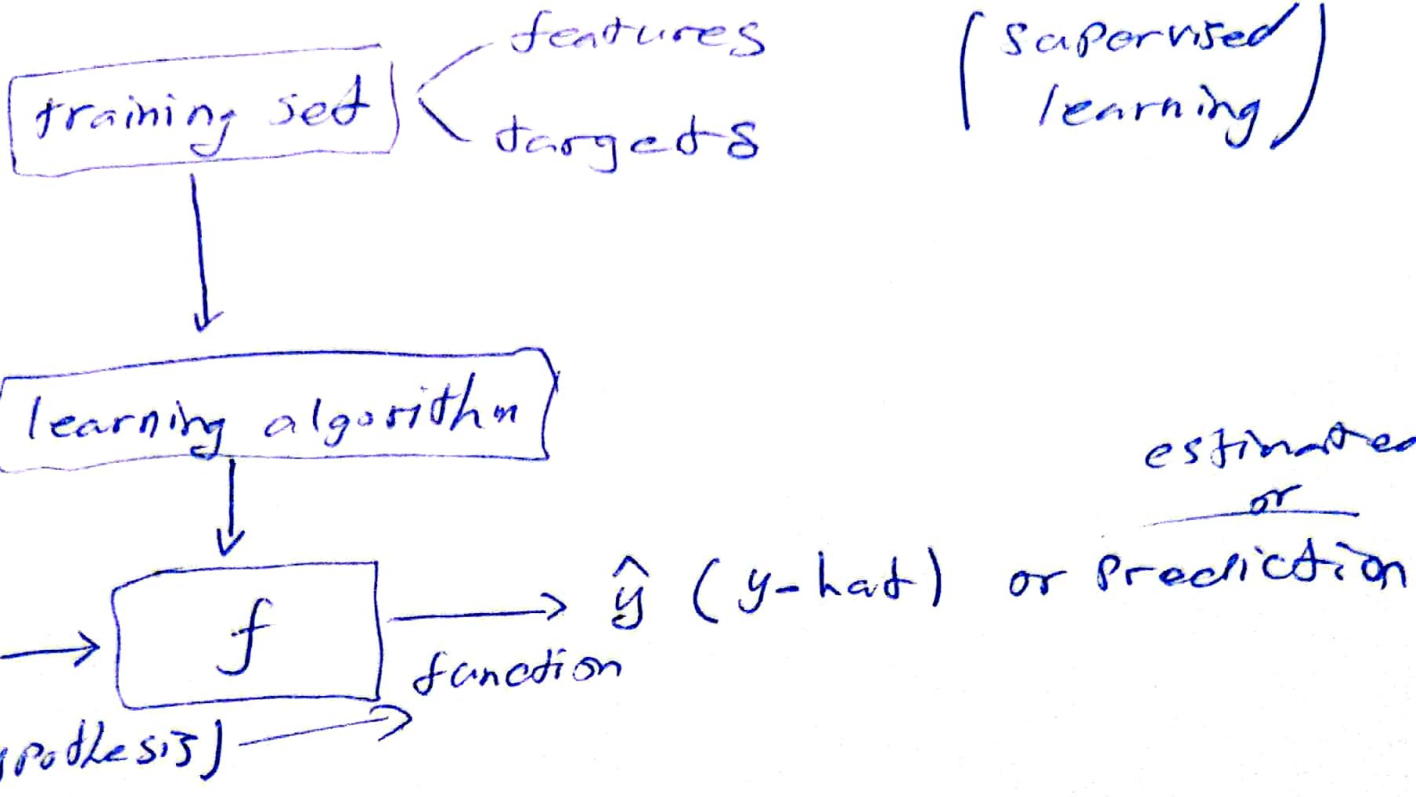
- m (lower case): number of training examples.

(x, y) = single training example.

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

↓ refers to a specific row in the table →

exp: $(x^{(1)}, y^{(1)}) = (2014, 400)$



feature → model → prediction

x (f) ŷ

How to represent "f"?

if f be a line:

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

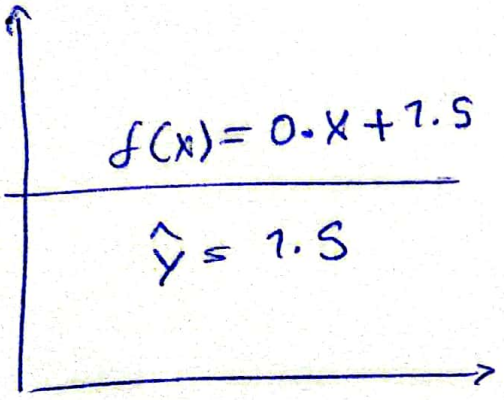
a linear regression with "one variable" (single feature)
↳ size

// "uni variable" linear regression

Cost function

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

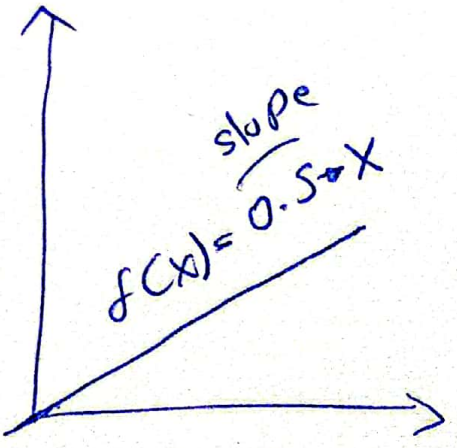
w and b are Parameters.
coefficients
or
weights



w = 0

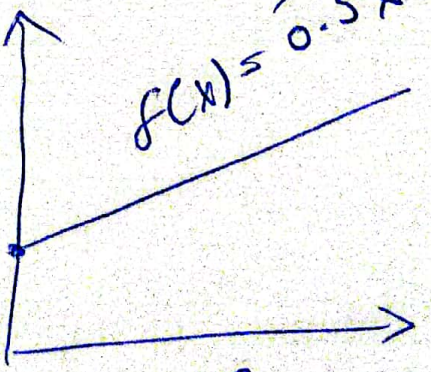
b = 7.5

intercept



w = 7.5

b = 0



w = 0.5

b = 7

$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

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

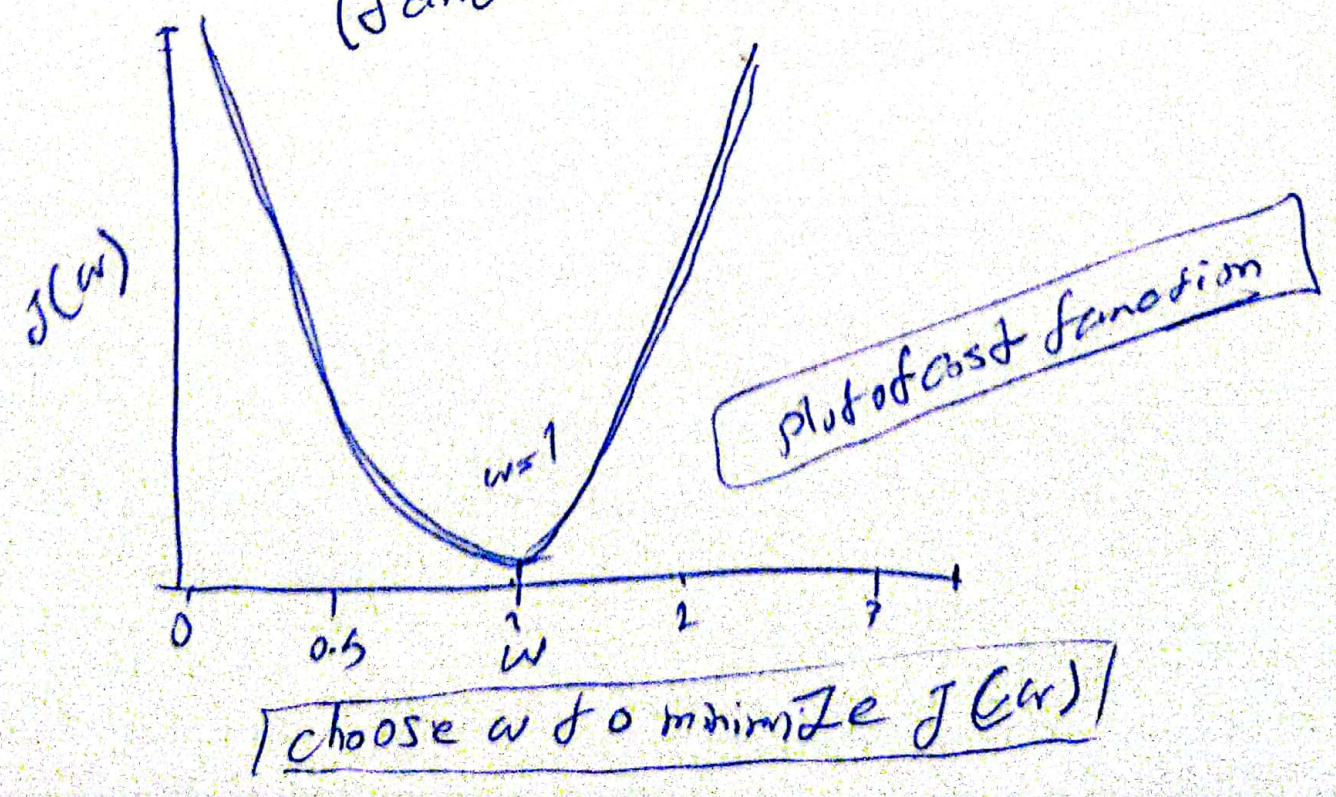
cost function: (squared error)

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

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

goal: $\min_{w,b} J(w,b)$

(function of w)



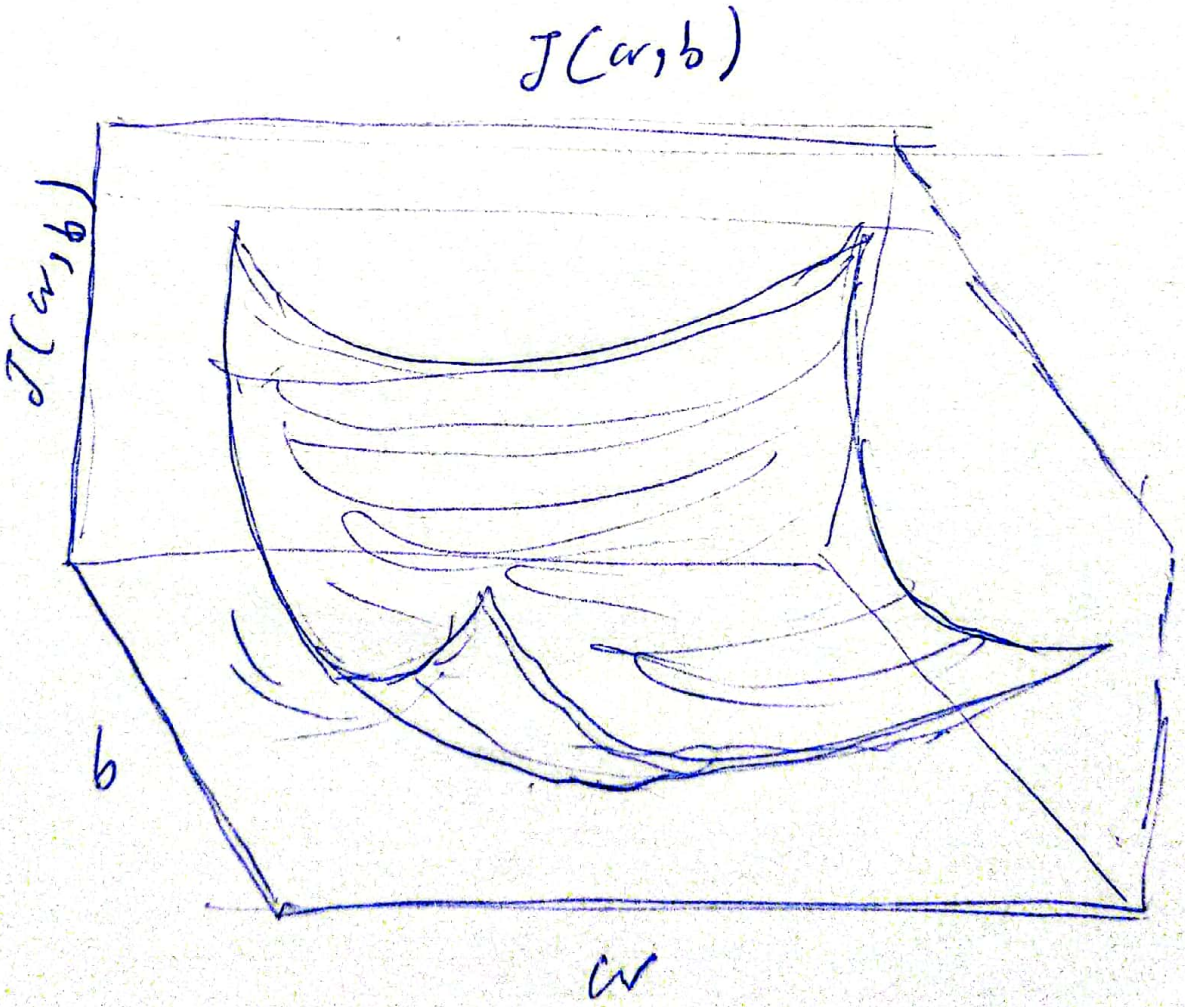
Visualizing the cost function

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

Parameters: w, b

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

Objective: $\underset{w,b}{\text{minimize}}\ J(w,b)$



Gradient Descent

cost function : $J(w,b)$

want to : $\min_{w,b} J(w,b)$

in general: $J(w_1, w_2, \dots, w_n, b)$

goal : $\min_{w_1, w_2, \dots, w_n, b} J(w_1, w_2, \dots, w_n, b)$

out line :

Start with some w, b (set $w=0, b=0$)

Keep changing w, b to reduce $J(w,b)$

until we settle at ~~the~~ or near a minimum.

may have > 1 minimum

Gradient Descent algorithm

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

α = learning rate (a small positive number)

$\frac{d}{dw} J(w,b)$ = derivative of cost function with respect to weight.

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

simultaneously update w and b

$$tmp\_w = w - \alpha \frac{d}{dw} J(w,b)$$
$$tmp\_b = b - \alpha \frac{d}{db} J(w,b)$$

↓

$$w = tmp\_w$$
$$b = tmp\_b$$

Gradient Descent is an algorithm for finding values of parameters w and b that minimize the cost function ($J(w,b)$)

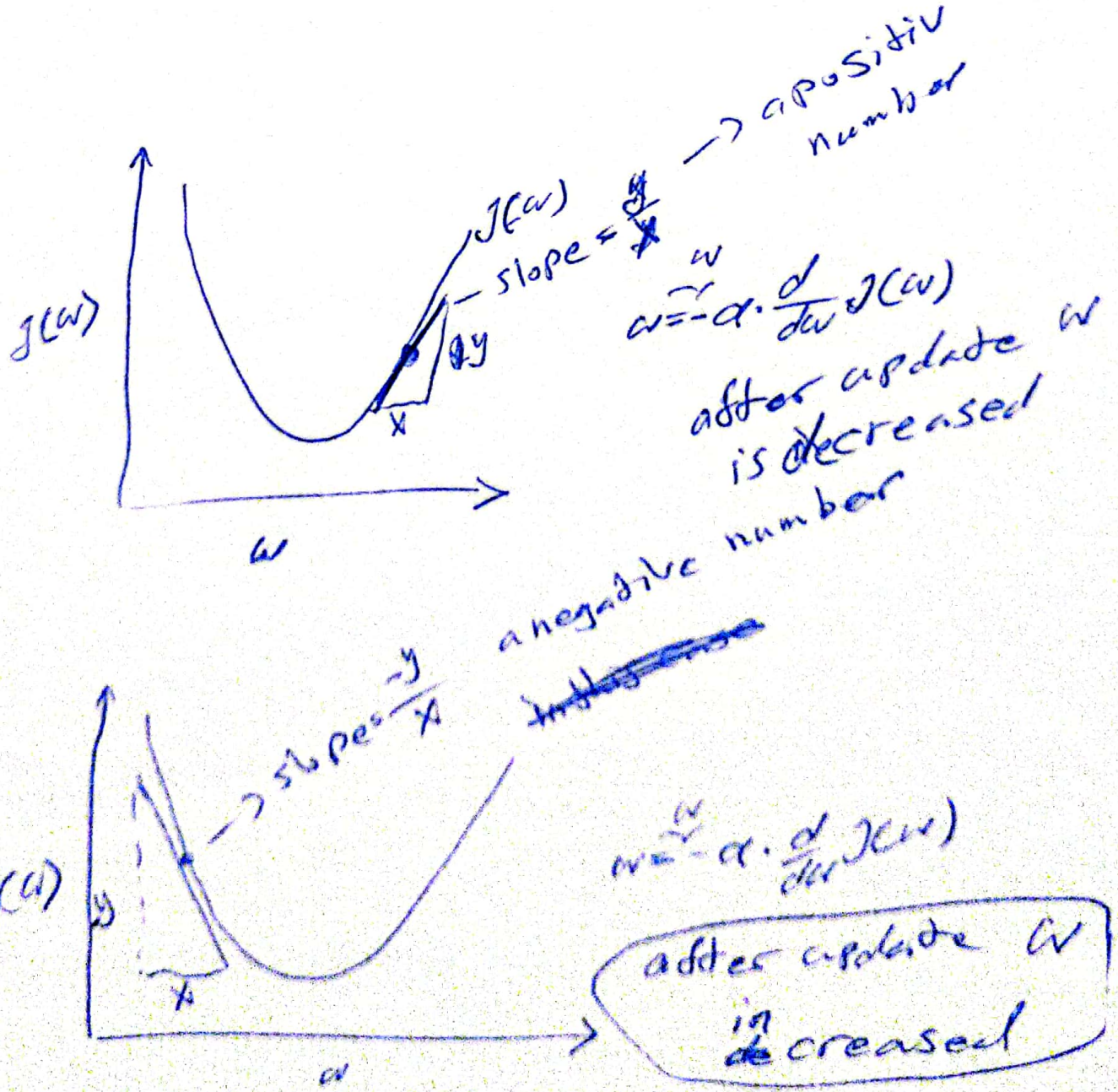
"Gradient Descent intuition"

repeat until convergence {

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

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

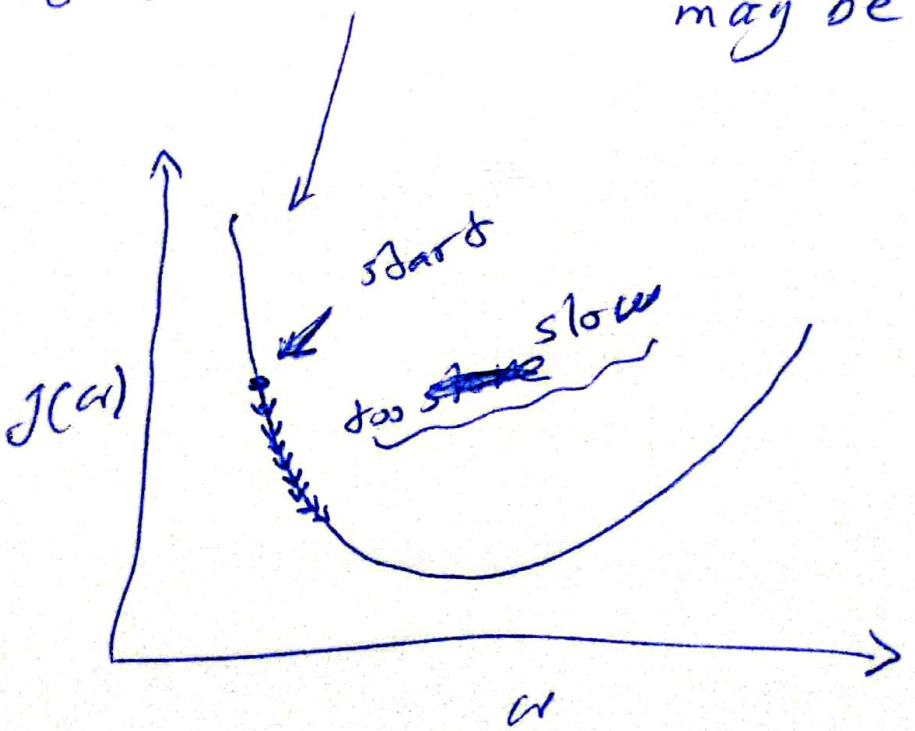
}



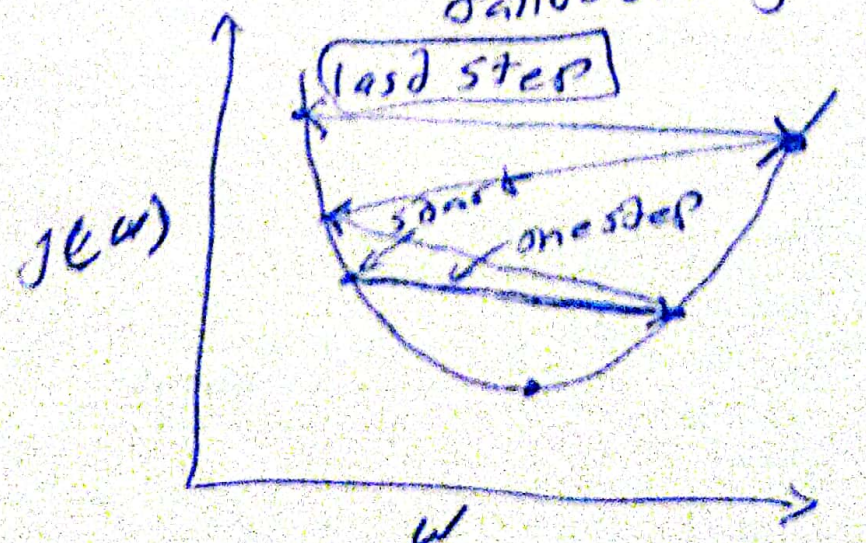
Learning Rate (α)

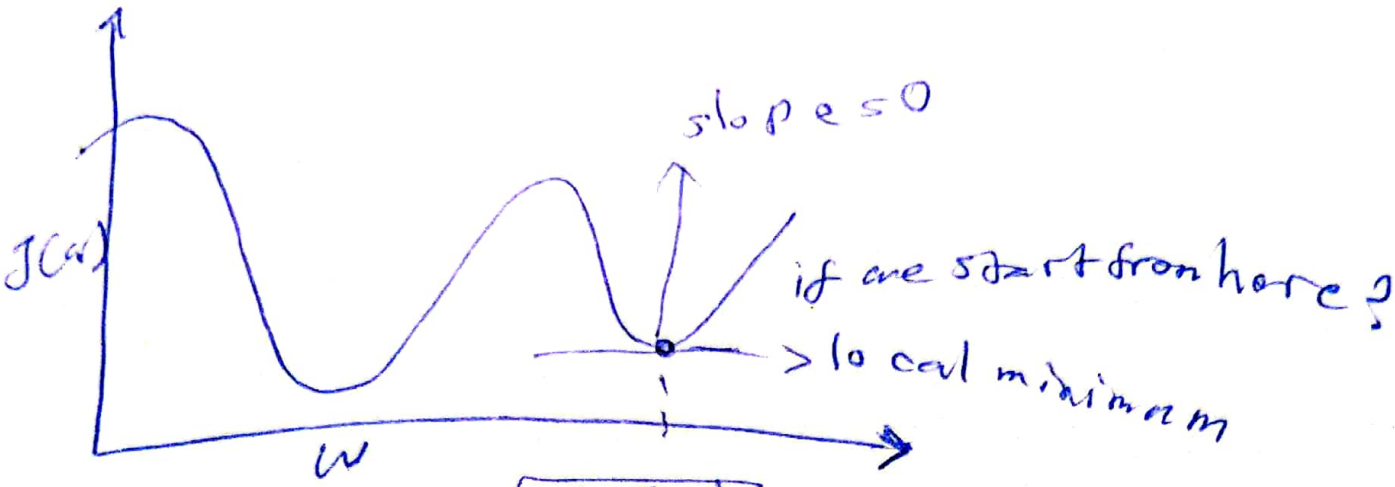
$$w = w - \alpha \cdot \frac{d}{dw} J(w)$$

if α is too small: Gradient descent may be ~~do~~ slow.



if α is too large: gradient descent may overshoot, never reach minimum. fail to converge or may diverge





$w = w - \alpha \cdot \boxed{\frac{d}{dw} J(w)}$ → $= 0$

so

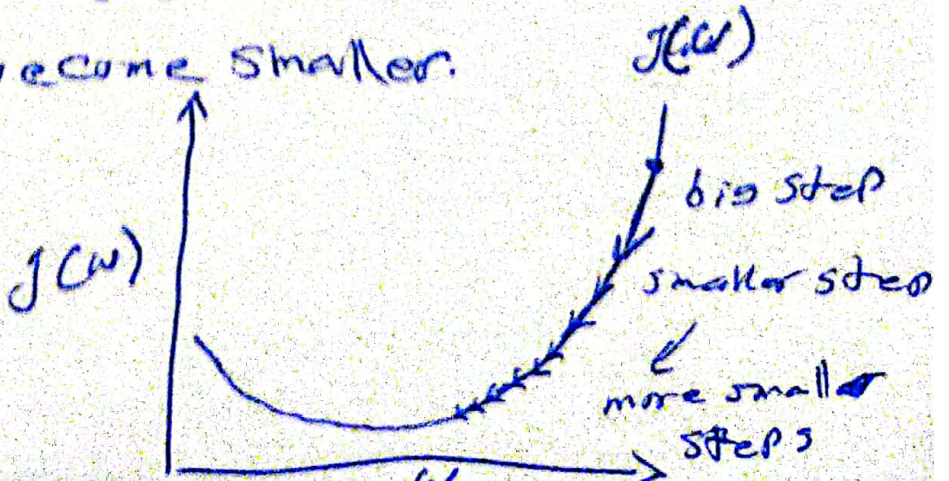
$w = w - \alpha \cdot 0$

so $w = w$

it means if we start from a local minimum gradient descent algorithm changes nothing

---

Even if α is a fixed number
updated steps change over iteration
near a local minimum
derivative becomes smaller
so update steps become smaller.



Gradient Descent for Linear Regression.

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

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

Gradient descent:

repeat until convergence {

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

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

}

Partial derivatives

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

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

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

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

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

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

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

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

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

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

Gradient Descent algorithm

repeat until convergence {

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

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

}

---

if you use the cost function with linear regression, (squared error cost)
it does not have multiple local minimum,
it has in this case just a "single"
"global minimum"

Batch gradient descent (normal gradient descent)
refers to use all training set to
update the parameters in each iteration.

multiple features (variable)

x_j = jth feature

n = number of features

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

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

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

Parameters:

$\vec{w} = [w_1, w_2, \ldots, w_n]$ a row vector of weights.

b is a number

$\vec{x} = [x_1, x_2, \ldots x_n]$ a row vector of features.

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

$\cdot$ ↑ dot product

↓

multiple linear regression

Gradient descent for multiple linear regression

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

$J(\vec{w}, b)$ ← cost function

repeat { Gradient descent

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

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

}

repeat {

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

$\vdots$

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

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

simultaneously update w_j (for $j = 1, \ldots, n$) and b

}

An alternative to gradient descent

→ normal equation

only for linear regression

solve for w, b without iteration.

disadvantages:

- does not generalize to other learning algorithms.
- slow when number of features is large ($> 10{,}000$)

Feature scaling

mean normalization:

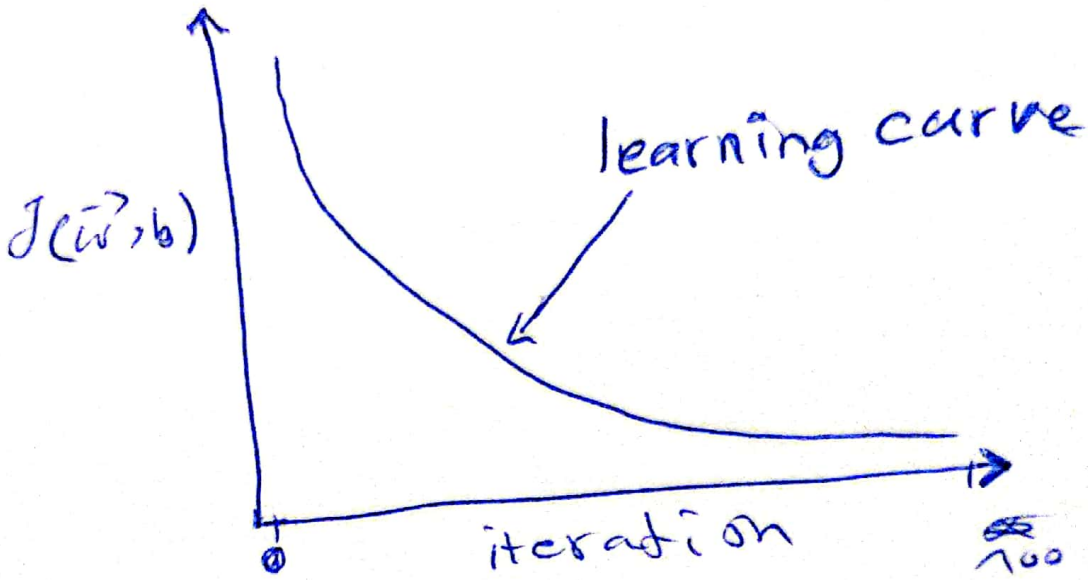
$$x_i = \frac{x_i - \mu}{\max - \min}$$

Z-score normalization

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

How to check if GD converges?

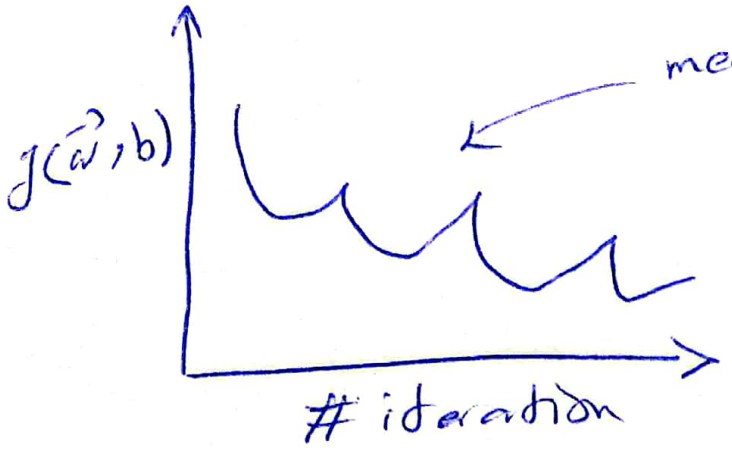
Plot the cost function:



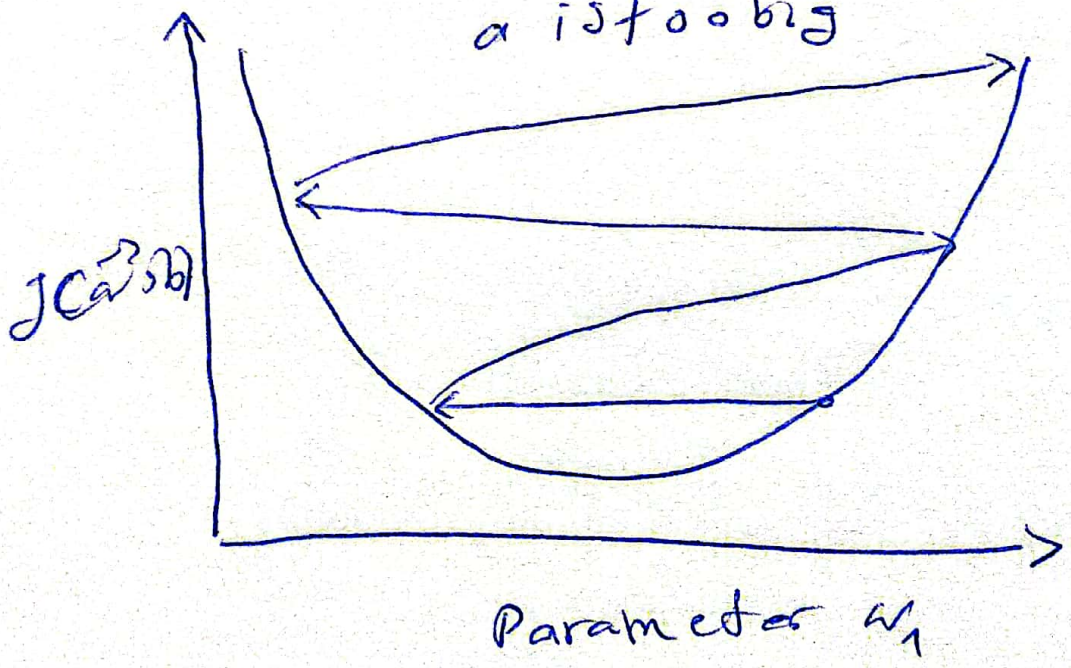
Automatic convergence test

if $J(\vec{w}, b)$ decreases by $\leq \varepsilon$ in one iteration, declare convergence.

Identify Problem with gradient descent

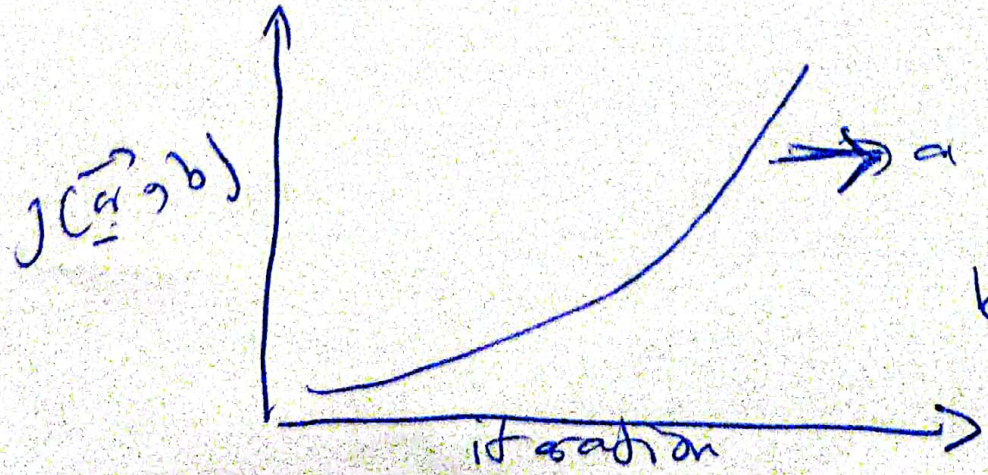


means
- code has bugs
or
- learning rate is too large.



α is too big

use a smaller learning rate.



α is too big
or
bug in the code

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

---

feature engineering

- create a new feature

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

$x_3 = x_1 \times x_2$ ← new feature

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

---

exp: price of a house

x_1 = frontage

x_2 = depth

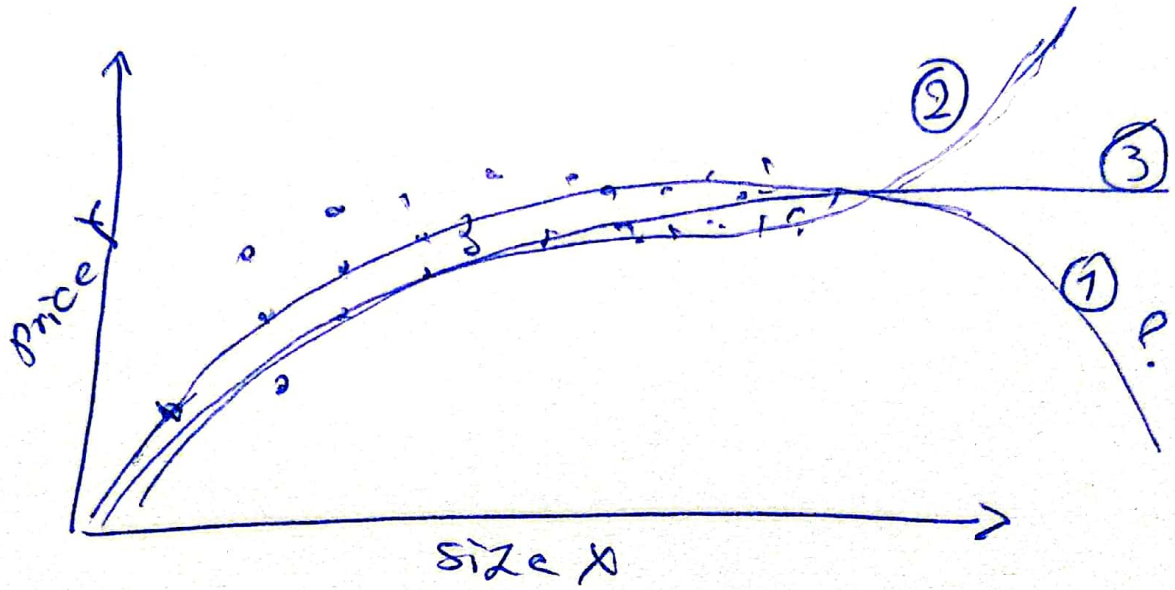
x_3 = area (frontage × depth)

---

feature engineering: using intuition to design new features, by transforming or combining

original features.

Polynomial regression



① quadratic function: $f_{(\vec{w},b)}(x) = w_1 x + w_2 x^2 + b$

not very appropriate ↗

② cubic function: $f_{(\vec{w},b)}(x) = w_1 x_1 + w_2 x^2 + w_3 x^3 + b$

a better model

↙ feature scaling really important because of "powers"

③ square root of $f_{(\vec{w},b)}(x) = w_1 x + w_2 \sqrt{x} + b$

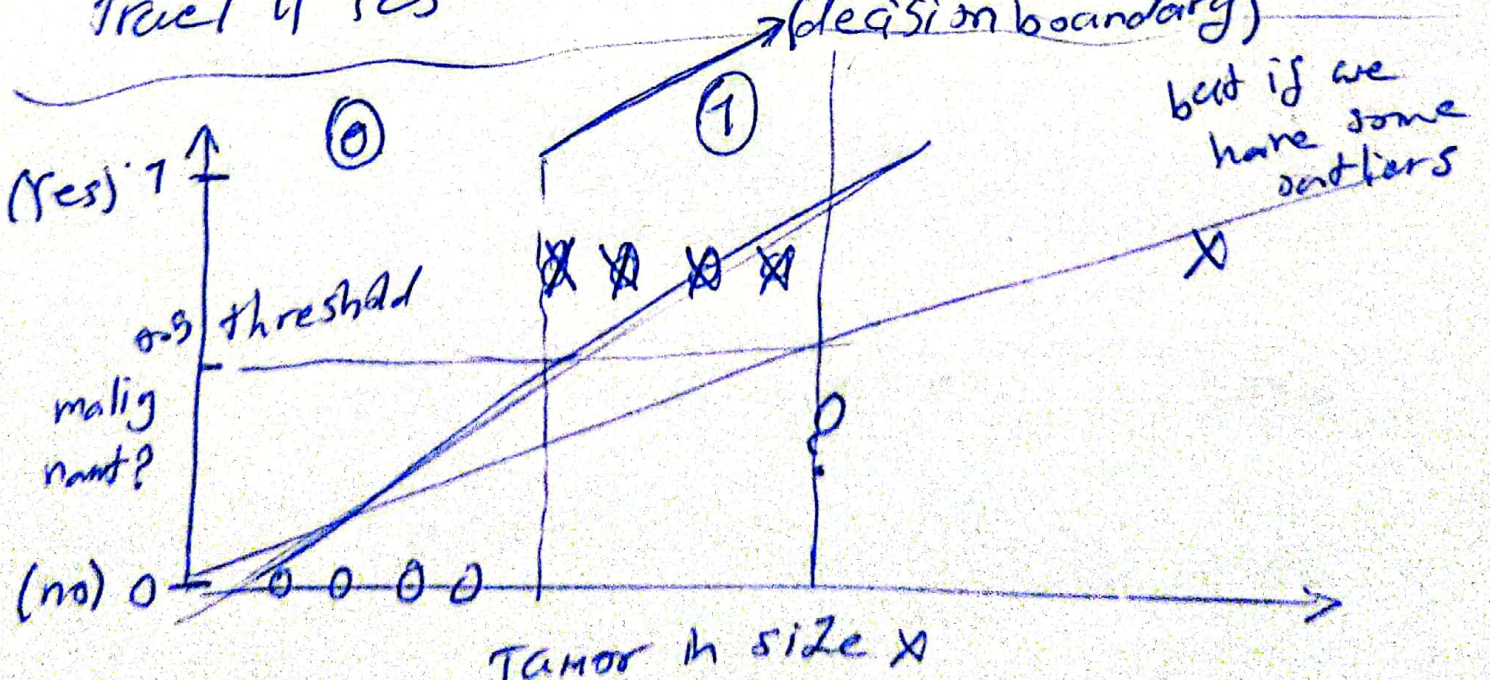
classification

Binary classification: when thy target (y) can be only categorized in two classes.

	1	0
Exps	~~true~~	~~false~~ false
- is this email spam?	yes	no.
- is the transaction fraudulent?	yes	no.
- is the tumor malignant?	yes	no.

false/~~zero~~0/ no → negative class.

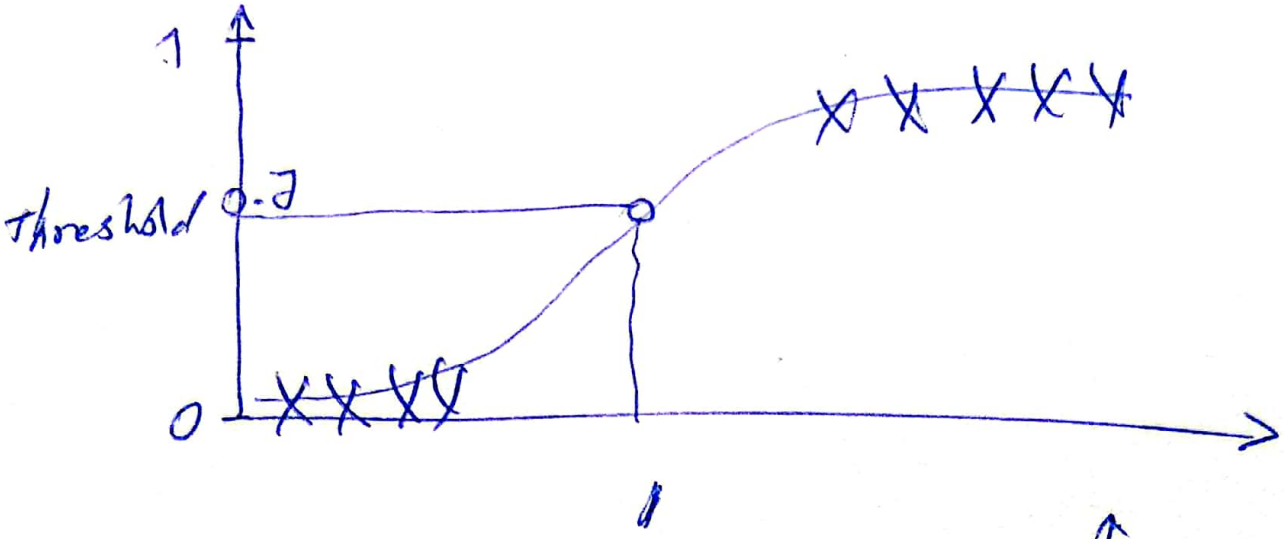
true/ 1/ yes → positive class.



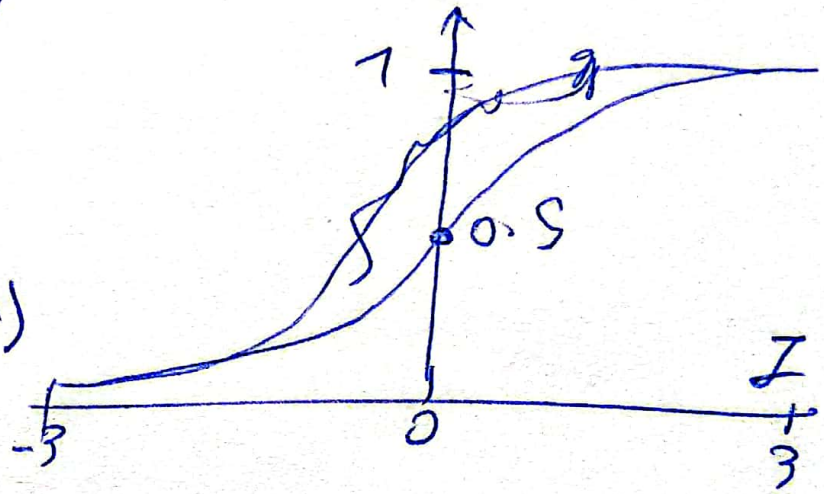
if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

logistic regression



sigmoid function
(logistic function)



out put between 0 and 1

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

when $z \gg 0$ | $g(z) \simeq 1$

when $z = 0 \rightarrow g(z) = 0.5$

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

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

$$\downarrow$$

$$z$$

$$\downarrow$$

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

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

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

The output of logistic regression is the "probability" of that class is 1

"probability" = number between 0 and 1

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

probability that y is 1,
given input $\vec{x}$, parameters $\vec{w}, b$

Decision boundary

logistic regression

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

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

$\downarrow$

z

$\downarrow$

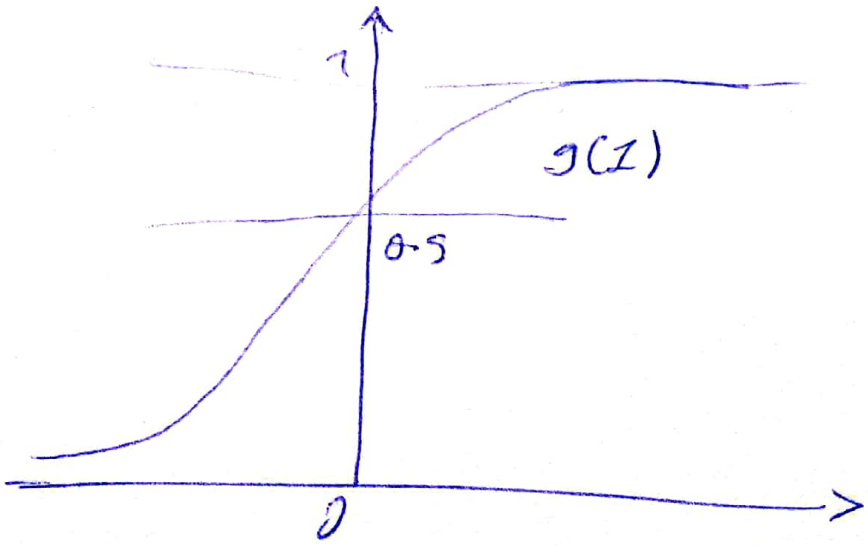
$g(z) = \frac{1}{1+e^{-z}}$

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

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

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

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



when is g of I greater or equal 0.5 ?

g of I is greater than or equal 0.5 whenever I is greater or equal ~~than~~ to Zero . it means ~~if~~ when I is 0 or Positive.

---

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

$g(I) \geq 0.5$

$I \geq 0$

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

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

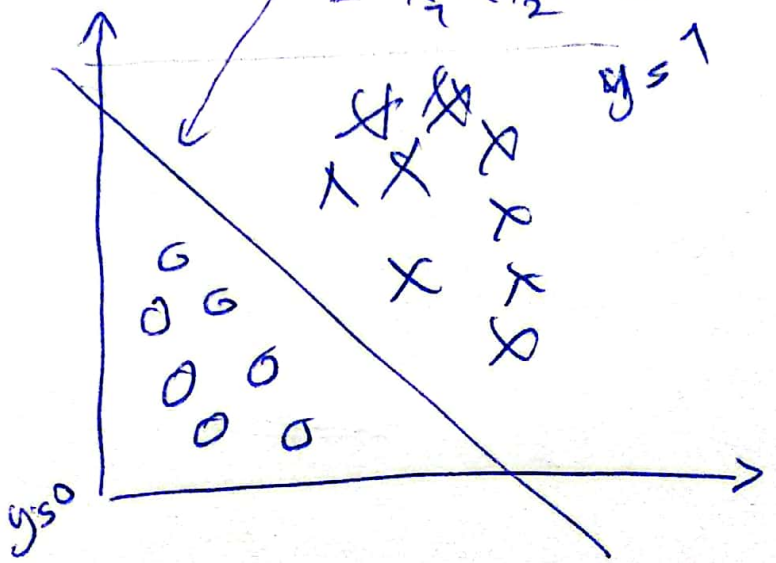
$\hat{y} = 1$ $\quad$ $\hat{y} = 0$

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

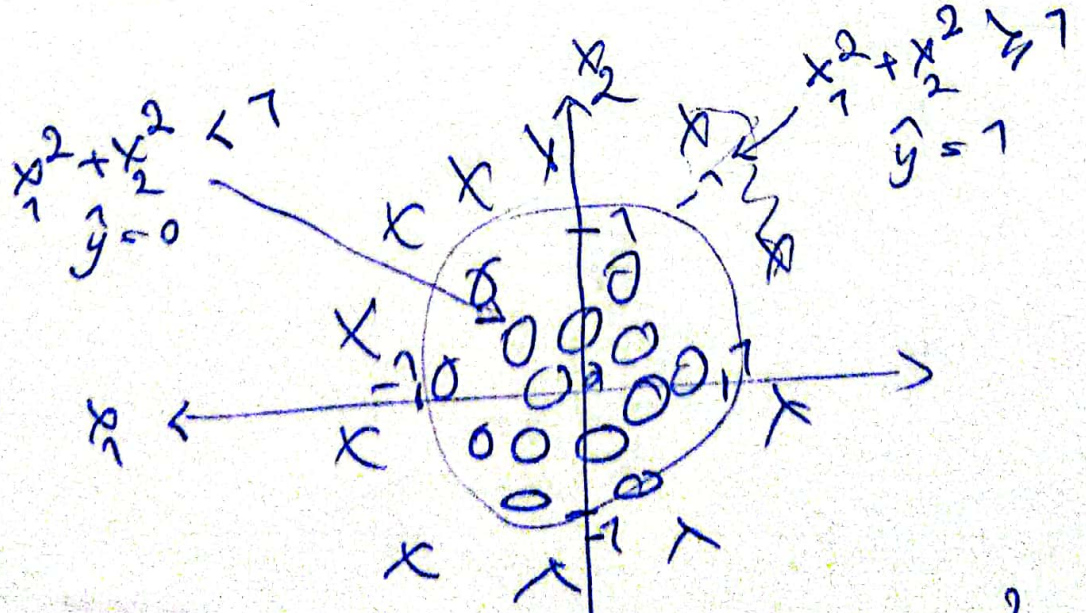
decision boundary: $z = \vec{w}\cdot\vec{x} + b = 0$

$z = x_1 + x_2 - 3 = 0$

$= x_1 + x_2 = 3$



non linear decision boundary



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

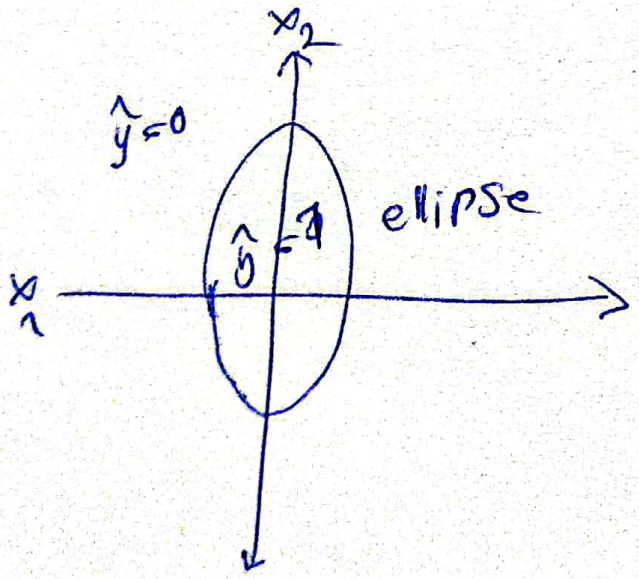
$$I = x_1^2 + x_2^2 - 1$$

decision boundary $= I = 0$

$$= x_1^2 + x_2^2 - 1 = 0$$

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

$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2 + \dots + b)$$



---

detection

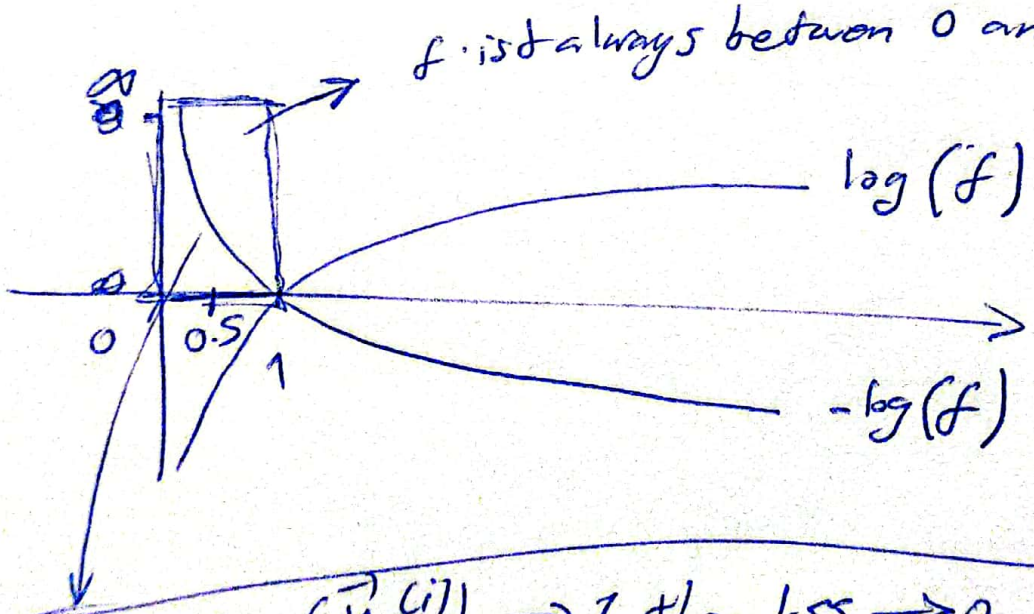
- let's say we are creating a tumor algorithm. our algorithm will be used to flag potential tumors for further inspection by a specialist. what ~~should~~ value should our we use for a threshold?

- ☐ high ex: 0.9
- ☐ low ex: 0.2

cost function for logistic regression

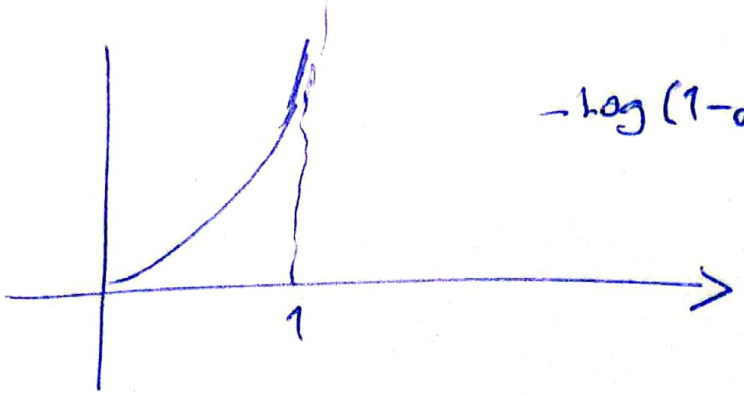
$$\text{loss} = L\left(f_{\vec{w},b}(\vec{X}^{(i)}), y^{(i)}\right)$$

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



As $f_{\vec{w},b}(\vec{X}_0^{(i)}) \to 1$ then loss $\to 0$

As $f_{\vec{w},b}(\vec{X}_0^{(i)}) \to 0$ then loss $\to \infty$



~~if $y^{(i)}=0$~~

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

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

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

if $y^{(i)}=1$:

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

if $y^{(i)}=0$:

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

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)\right]$$

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

Training logistic regression

find $\vec{w}, b$ (optimal)

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

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

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

repeat {

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

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

}

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

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

"Gradient descent for logistic regression"

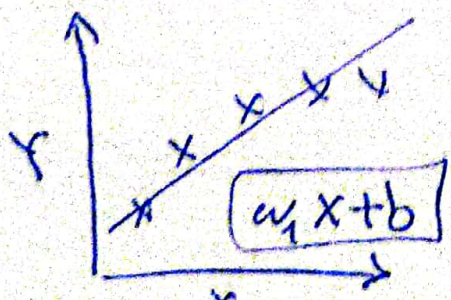
repeat {

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

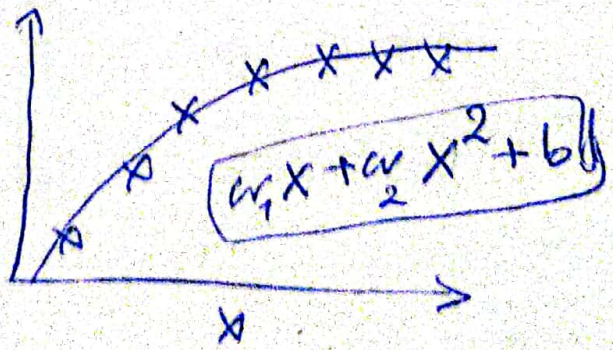
$$b = b - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right) \right]$$

} simultaneous updates

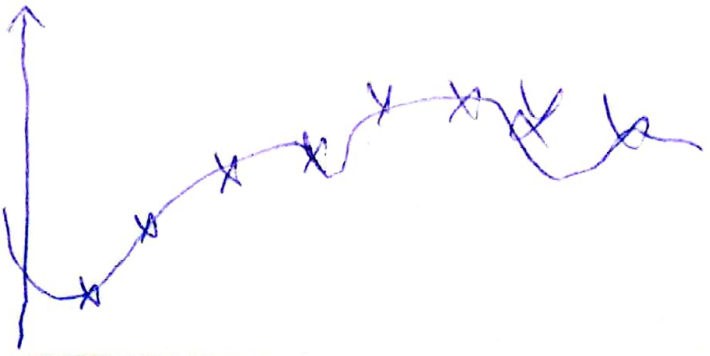
Overfitting



(underfit)
or
(high bias)



generalization



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

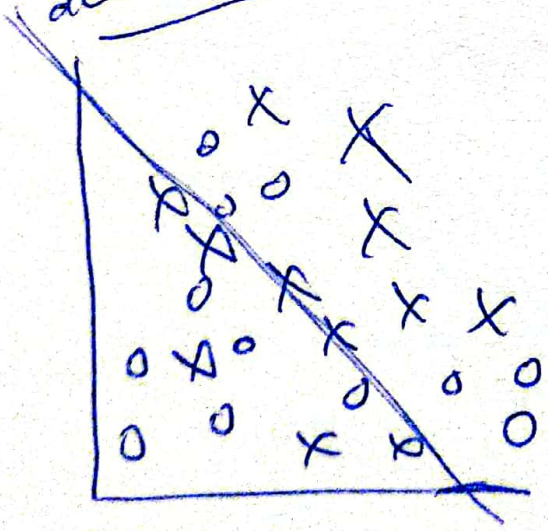
the model is overfitted

cost $\simeq 0$ but

The model has (overfitting problem)
or
(high variance)

decision boundary

misclassification

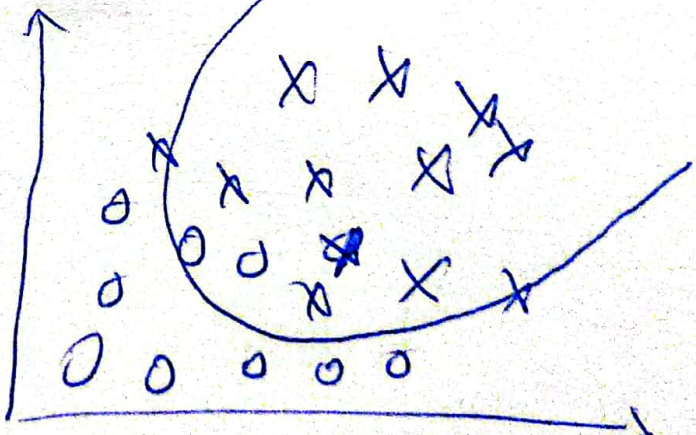


$z = w_1 x_1 + w_2 x_2 + b$

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

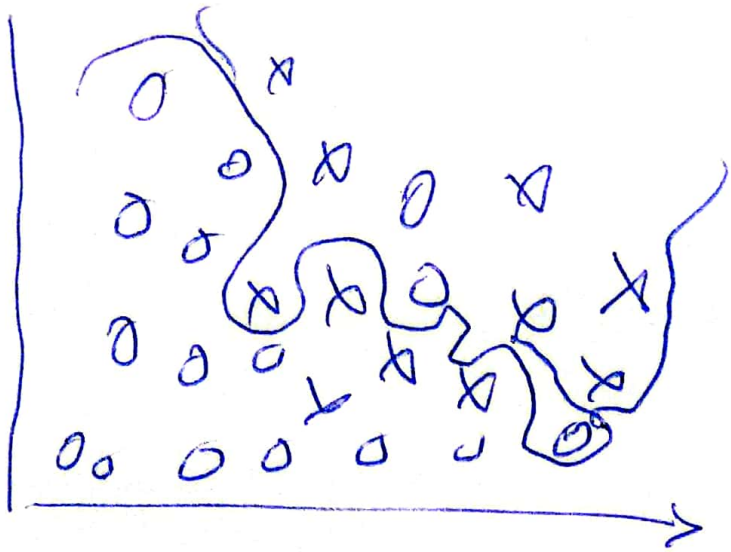
underfit
or
high bias

decision boundary



$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + b$

fit



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 x_2 + w_4 x_1^2 x_2^2 + w_5 \cdots + b$$

→ overfit

---

if the model has overfitting

what we should do?

1. collect more training examples
2. Select features to include/exclude (feature selection)
3. regularization (reduce the impact of some features)

$$f(x) = 13x - 0.23x^2 + 0.00074x^3 - 0.00007x^4 + b$$

small values for w_j

cost function with regularization

model: $w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^9 + b$

if we want to regularize w_3 and w_4:

~~cost function~~

$$J_{\vec{w},b}(\vec{x}) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \underbrace{1000 w_3^2 + 1000 w_4^2}_{\text{a large number}}$$

~~cost~~

if we have a model with a high number of features (eg: 100 features) we can regularize all of them.

So:

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

sometimes: $+ \frac{1}{2m} b^2$

n = number of features.

m = number of data points.

λ (lambda) = regularization Parameter — a big number ~~or small number~~

if lambda is 0 ⇒ model is underfit.

if lambda is too ~~big~~ large model is overfit.

---

regularized linear regression

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

Gradient descent {

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

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

}

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

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

if we regularize b's

$$+ \frac{1}{m} b$$

Gradient descent with regularization

GD {

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

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

} simultaneously update.

Regularized logistic regression

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

GD

repeat {

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

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

}

(47)

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

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