ML Defn: "Anthun Samuel" (1959) → " Field of study that gives computers ability to Learn without being explicitly programmed

Supervised (Learner given right Answer) → x → y
'input' 'output' Lebel

Regression → predict a number [x:y mapping]
- email → spam
- audio → text transcript
- image and nadden → position of other cars

Classification
0 c.m — 10 c.m

Age
Tumon size

Unsupervised Learning → unlebeled data
- age
- tumonsize
- clausteping algorithm → DNA Micnoarray
- grouping customen
- samillian Data point together

Dimentional Reduction → compress data to fewen numbens

Anomaly detection → Find unusual Data point

Terminology

Training set : Data used to trained the model

x → input variable

y → output variable / target variable

training set → features / target

learning algorithm

$x \rightarrow f \rightarrow \hat{y}$ (function)

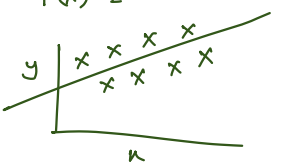
feature → model → prediction (estimated y)

size → f → price

How to represent:

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

$f(x) =$



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

$f(x) = wx + b$

linear regression with one variable

Cost Function:

w, b do → parameter

$\frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$ error = $J(w,b)$ → squared error cost function

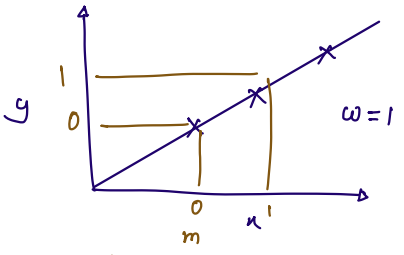
m = number of training examples

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

{ w, b → can be adjusted }

Cost Function Intuition:

$f_w(x)$
↳ For fixed w, function of x



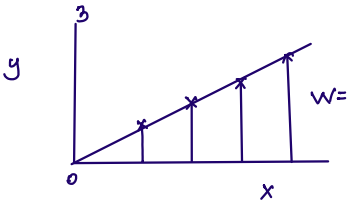
$$J(1) = \frac{1}{2m}\sum_{i=1} f(1)(x^i - y^i)^2$$

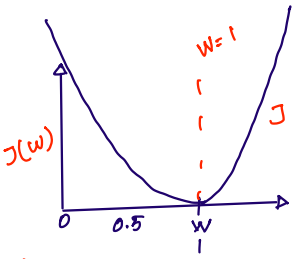
$$= \frac{1}{2m}(0 + 0 + 0)$$

$$= 0$$

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

$$= \frac{1}{2\times 3}[3.5] = 0.58$$



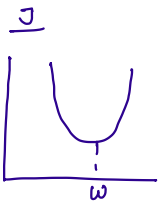


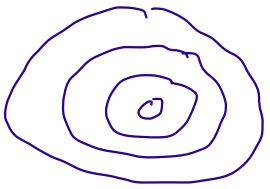
Goal → $\min_w J(w)$

general case → $\min_{w,b} J(w,b)$

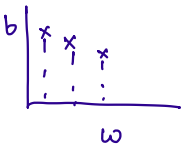
f w, b

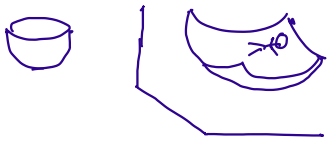
cost function





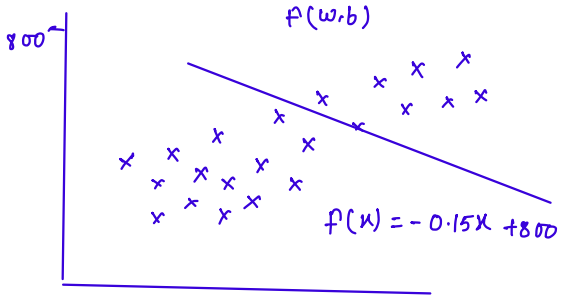
J(w,b)

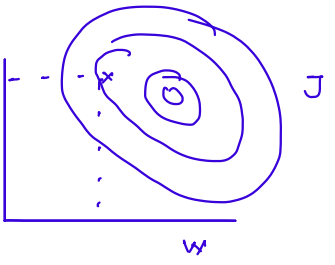


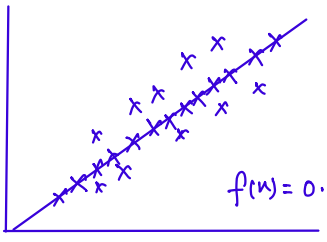


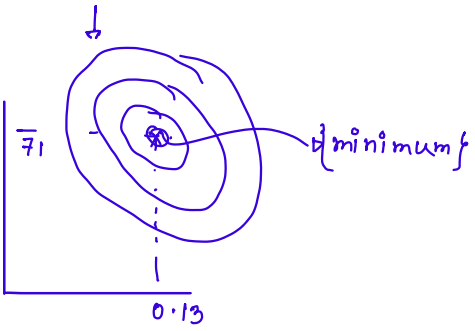
J(w=-10, b=15)

* contour plot









- Best fit line → Manually not possible

cost function,

$$J(w,b) = \frac{1}{2m} \sum_{i=0}^{m-1} f(w,b(x^i) - y^i)^2$$

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

Gradient descent

↳ Not bound in ML also in DL

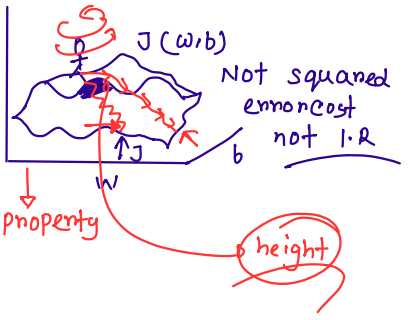
for linear regression

$\min_{w,b} J(w,b)$ → To minimize

start with some w, b

↓ set (w=0, b=0)

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



Training L.R

Implementing Gradient Descent

Gradient Descent Algorithm:

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

→ Derivative term

Assignment operator

Learning rate

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

two update take place

* simultaneously update w, b

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

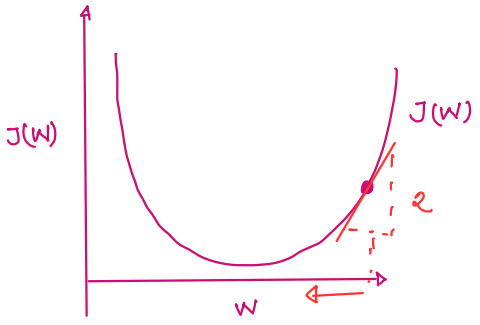
$$temp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

Not partial → Here full

↳ Mathematical

$w = temp\_w$

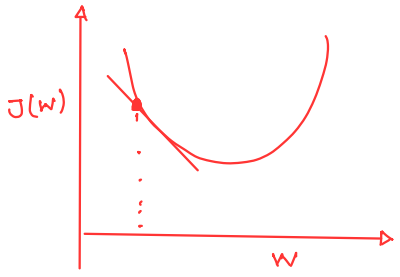
$b = temp\_b$



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

$w = w - \alpha$ (positive number)

Decrease



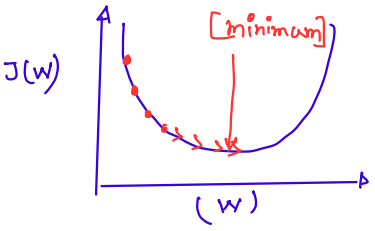
$\frac{d}{dw} J(w) < 0$

$w = w - \alpha$ (negative number)

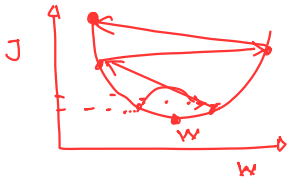
Increase

Learning Rate

If α is too small...

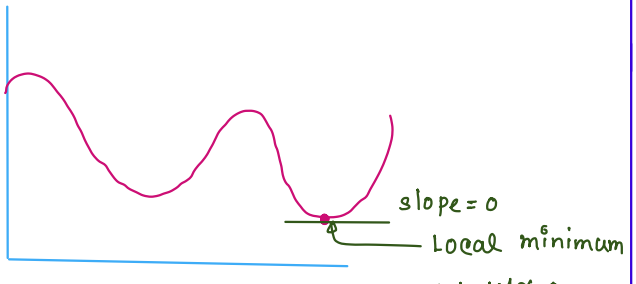


too Large:



Gradient descent may:

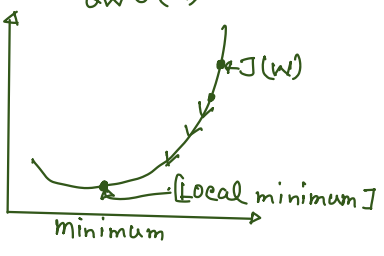
- Overshoot
- Never reach minimum



$w = w - \alpha \cdot 0$

$w = w$

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



Regression with multiple input variable

$f_{w,b}(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

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

b is a number

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

multiple linear regression: $f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = w_1x_1 + w_2x_2 + \dots + w_nx_n + b$

($\cdot$ dot product)

Vectorization

Makes code faster!

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

#vectorization:

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

$j = 1 \dots n$ (1, 2, 3)

```
f = 0
for j in range(0,n):
    f = f + w[j] * x[j]
f = f + b
# Without vectorization
```

Gradient Descent:

$w_1 = w_1 - 0.1d_1$

$w_2 = w_2 - 0.1d_2$

$\vdots \quad \vdots$

$w_n = w_n - 0.1d_n$

```
for j in range(0,n)
  w[j] = w[j] - 0.1 * d[j]
```

Without vectorization

$w = w - 0.1d$ → with vectorization

Normal equation → alternn method

↳ For linear regression

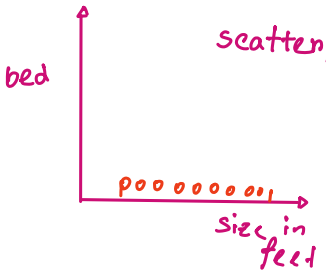
gradient descent → recommended Method

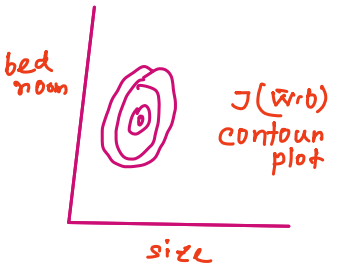
Feature and Parameter value

price = $w_1 x_1 + w_2 x_2 + b$

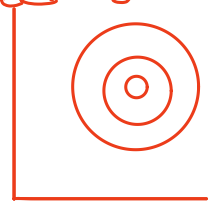
x_1: size, x_2: bedrooms

→ Features





after gradient descent Algorithm



Feature scaling

goal → make range small

$3000 \le x_1 \le 2000$

$x_1 \text{ scaled} = \frac{x_1}{2000}$

values (0-1) (-1-1) centered 0 with st. dev. 1

Mean Normalization:

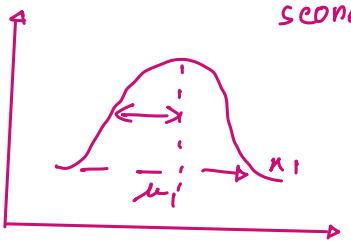
$x_1 = \frac{x_1 - \mu_1}{2000 - 300}$

$-0.18 \le x_1 \le 0.82$

Z score Normalization

Purpose: center the Data

standard score



$\sigma_1 = 950$

$\sigma_2 = 1.9$

standard deviation

$x_1 = \frac{x_1 - \mu_1}{\sigma_1}$

$-0.67 \le x_1 \le 3.1$

Feature scaling aim for about $-1 \le x_j \le 1$ for each feature

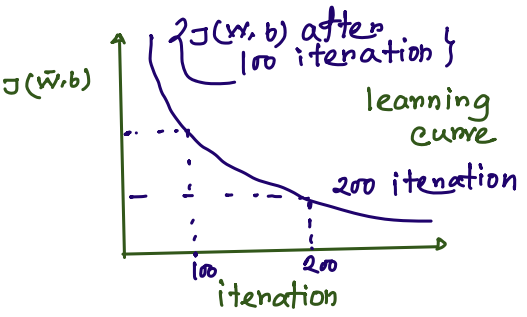
$-3 \le x_j \le 3$

$-0.3 \le x_j \le 0.3$

Make sure gradient descent is working correctly.

Objective : min $J(\vec{w}, b)$

$J(\vec{w}, b)$ should decrease after every iteration



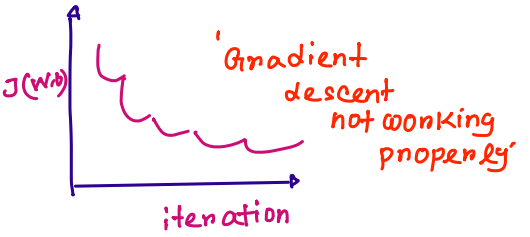
Automatic convergence test:

Let ε be 10^{-3} 0.001

$J(\vec{w}, b)$ decreases by $\le \varepsilon$ in one iteration.

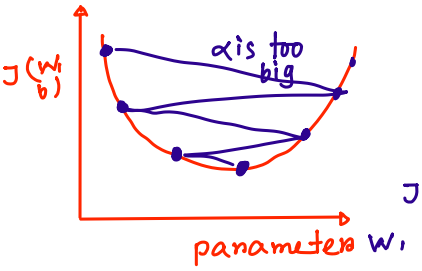
declare convergence

"Choose Learning Rate"

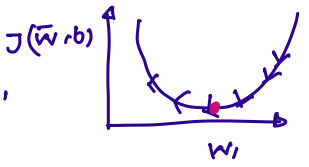


Adjusting Learning Rate

choosing Learning Rate:



smaller Learning rate



$w_1 = w_1 + \alpha d_1$ X

$w_1 = w_1 - \alpha d_1$ ✓

values of α try to 0.001, 0.01, 0.1, 1 ···

$J(\vec{w}, b)$, α should decrease on every iteration

Feature Engineering

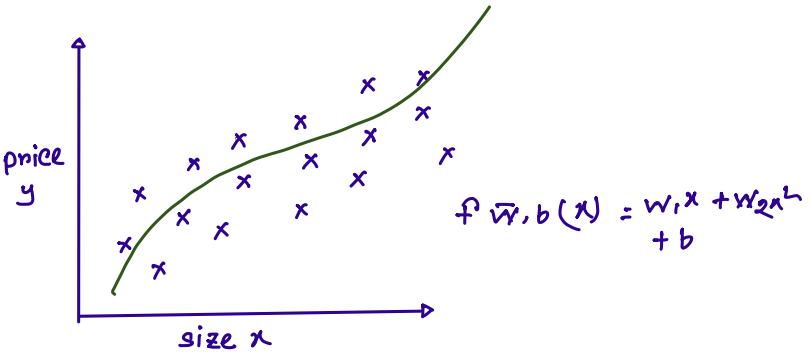
$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + b$

transforming, combining original features.

area = frontage × depth

$x_3 = x_1x_2$ → new feature

Polynomial Regression



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

Classification

yes/no — False/true

Binary Classification

0 → neg. Class

1 → Pos. Class

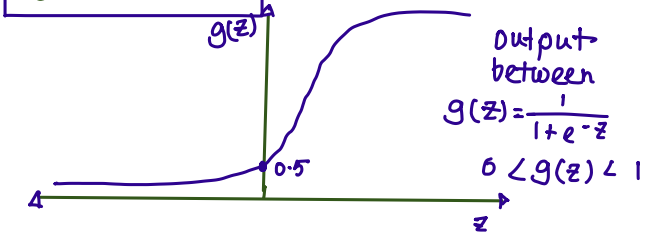
useful for classification

[Class - catagory]

- desecision boundary

{Logistic Regression}

Sigmoid Function:



output between

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

$0 < g(z) < 1$

(*)

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

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

Probablity:

[# $P(y=0) + P(y=1) = 1$]

Desecision Boundary

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

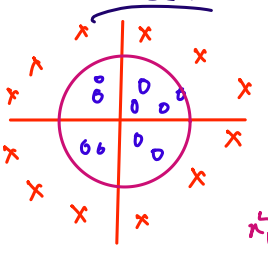
Desecision Boundary

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

$z = x_1 + x_2 - 3 = 0$

$x_1 + x_2 = 3$

Non Linear Desecion Tree:

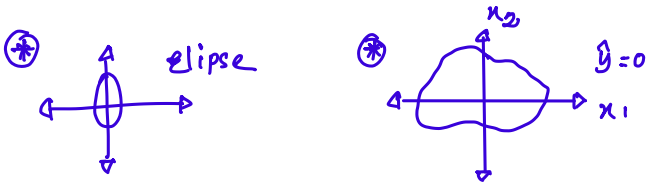


$z = x_1^2 + x_2^2 - 1 = 0$

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

$= g(w_1x_1^2 + w_2x_2^2 + b)$

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

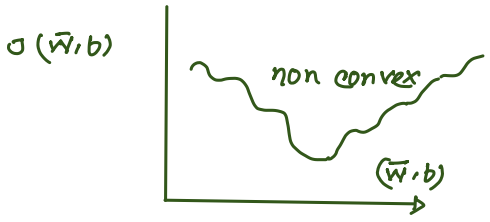


Cost Function

target 0,1

$J(\vec{w}, b)$ convex → Linear Regression

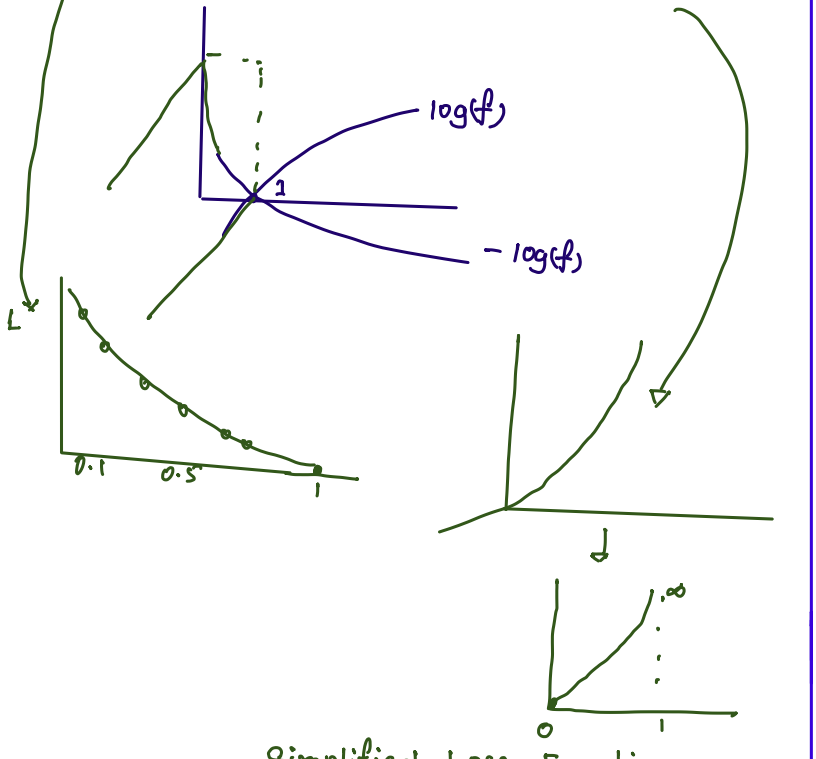
logistic,



logistic Loss function:

can reach global convex min.

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



Simplified Loss Function

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

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

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

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

$Lf_{\vec{w},b}(\vec{x}^{(i)}, y^{(i)}) = \ldots$

Simplied cost Function

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

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

1 iteration – loss → 1 ennon

average loss - cost ennon

Gradient descent

- cost minimum

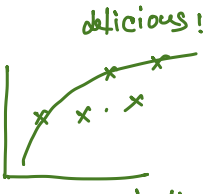
Formula same as linear regression

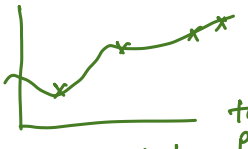
↓

significance Different ☺

* Overfitting !!! ☹

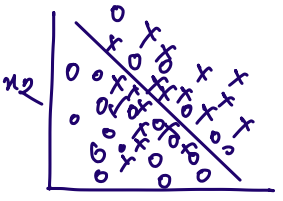
Reg.

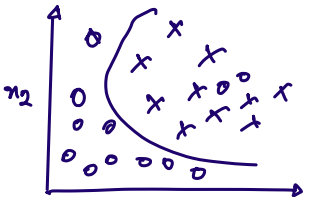


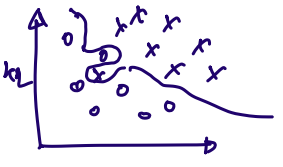


high variance

Class.







Regularization:

↓

λ → Regularized term

$\lambda > 0$

#Intuition

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

{mean squared ennon}