2:

(1)

Linear Regression (Multivariate)

x_1	x_2	...	x_n	y

i/p: $x_1, x_2, \ldots, x_n$ → output: y

Real values.

Task: Given $(x_1, x_2, \ldots x_n)$ we need to predict the value of y.

Eg:

- predict mpg value given engine parameters.
-

cylinder	HorsePower	Height	Weight	...	MPG
22.0	8	390	190	...	?
x_1	x_2	x_3	x_4	x_d	

? → predict mpg of a vehicle.

N = #No. of instances = 398 (rows).

Training Data → Learning Algo → Model

$(23, 8, 391, 120, \ldots)$ i/p → Model → mpg = 29.75 predicted value.

Hypothesis fn / Regression fn.:

$$\hat{y}_i(x) = b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_d x_{id}$$

x_1	x_2	...	x_d	y

Data set.

$x_1 = (x_{11}, x_{12}, \ldots x_{1d})$.

$x_2 = (x_{21}, x_{22}, \ldots x_{2d})$

- Need to find $(b_0, b_1, b_2, \ldots, b_d)$ parameters?
- Use training data and learning algorithm to compute these parameters.

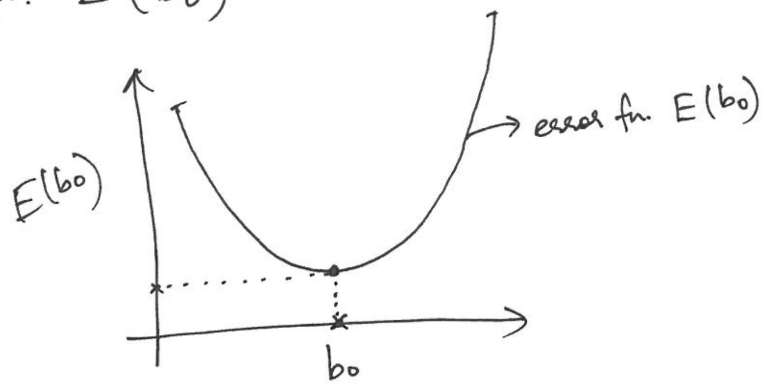
cost fn.?

$$E(b_0, b_1, \ldots b_d) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

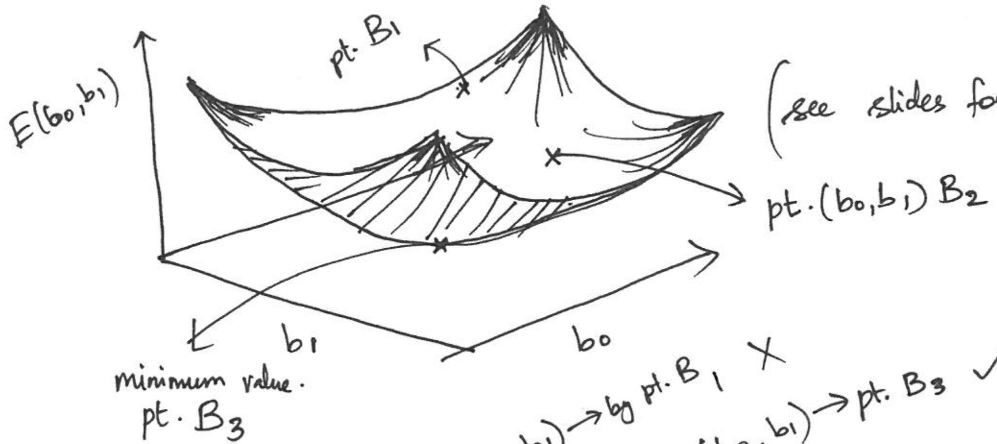
Goal minimize $E(\cdot\cdot)$ for different values of $(b_0, b_1, \ldots b_d)$

- squared error b/w y_i (from training data) and $\hat{y}_i$ (from predicted data using param $b_0, \ldots, b_d$).
- squared error cost fn.

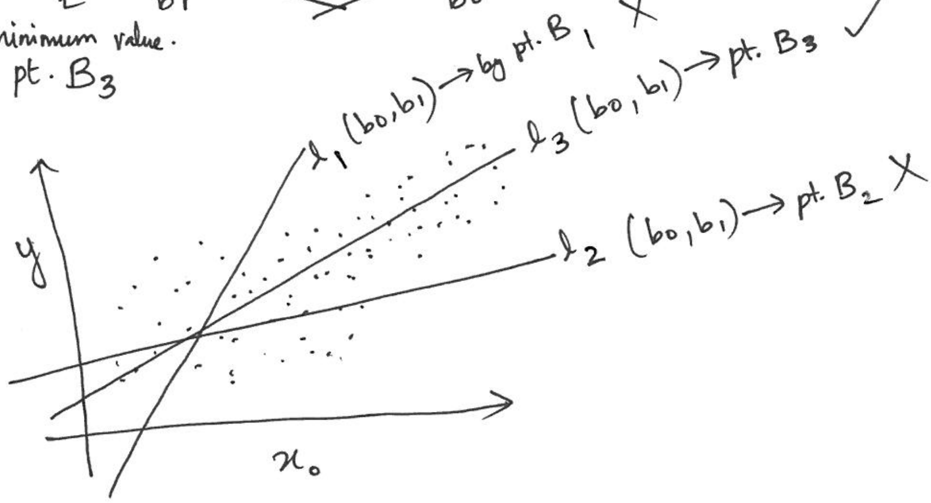
Cost / Error fn: $E(b_0) \longrightarrow$ 1 param.



$E(b_0, b_1) \longrightarrow$ 2 param.



(see slides for this figure)



$E(b_0, b_1, \ldots b_d) \longrightarrow$ d - param.

Goal $\rightarrow$ $\underset{(b_0, b_1 \ldots b_d)}{\text{minimize}}\ E(b_0, b_1 \ldots b_d)$

$\downarrow$

learning Algorithm ?

$\boxed{\text{Gradiant Decent}}$

3

Gradiant Decent

Outline :

1. start with some (b_0, b_1)

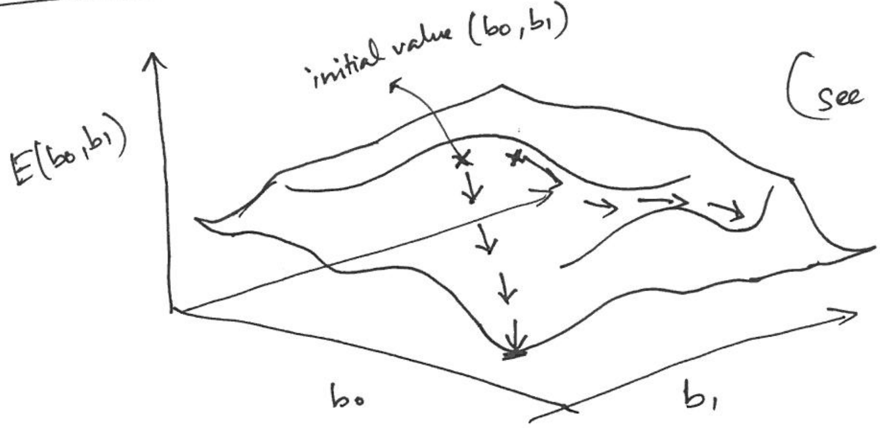
or $b_0 = 0$, $b_1 = 0$

2. Keep changing (b_0, b_1) value to reduce $E(b_0, b_1)$, until we end up in min. value

given $E(b_0, b_1)$

Goal $\rightarrow \min_{(b_0, b_1)} E(b_0, b_1)$

you can Extend it to $(b_0, b_1, \ldots b_d)$



(see slides for this figure)

Math :

repeat until convergence { (no change in b_j values.)

$a := b \rightarrow$ assignment.

$$b_j := b_j - \alpha \frac{\partial}{\partial b_j} E[b_0, b_1]$$

(for $j = 0$ and $j = 1$)

}

α: learning rate or update rate

$\frac{\partial}{\partial b_j}$: partial derivative of E w.r.t b_j

simultaneous update

{ $tmp0 := b_0 - \alpha \frac{\partial}{\partial b_0} E(b_0, b_1)$

$tmp1 := b_1 - \alpha \frac{\partial}{\partial b_1} E(b_0, b_1)$

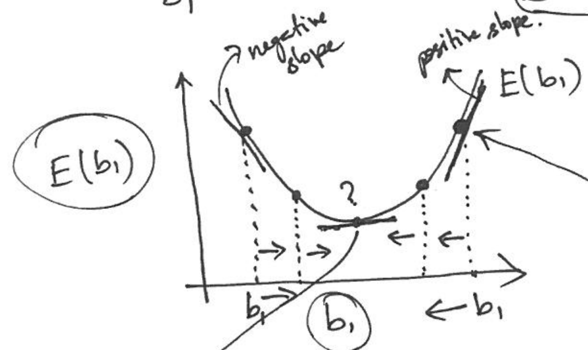
$b_0 := tmp\,0$

$b_1 := tmp\,1$

}

(4) **Example:** 1-param $E(b_1)$

$$\min_{b_1} E(b_1) \qquad b_1 \in \underset{\text{Real numbers.}}{R}$$



$$b_1 := b_1 - \alpha \overbrace{\underbrace{\frac{d}{db_1} E(b_1)}_{\text{slope of tangent to the fn. at the point } b_1}}^{\text{rate of change of } E \text{ at } b_1.}$$

$b_1 := b_1 - \alpha$ (positive value)

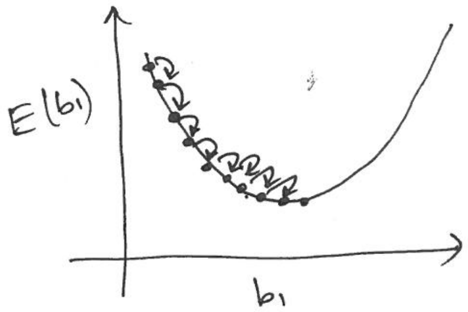
$b_1 := b_1 - \alpha$ (negative value)

slope at this pt?

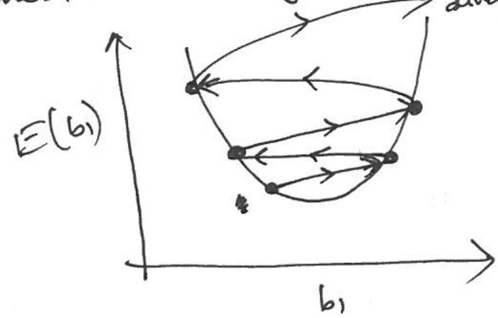
$\frac{dE}{db_1} = 0.$

α-value: (0 to 1) Eg: 0.001 $\rightarrow$ trial-and-error.

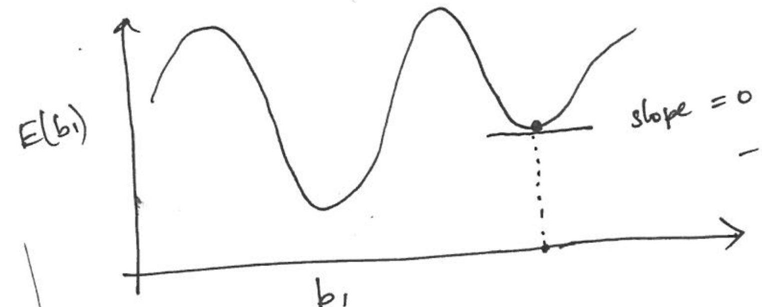
if α is too small, GD can be slow.



- if α is too large, GD can overshoot minimum.
- if may cause GD to fail to converge.
- Some time it may cause GD to diverge.



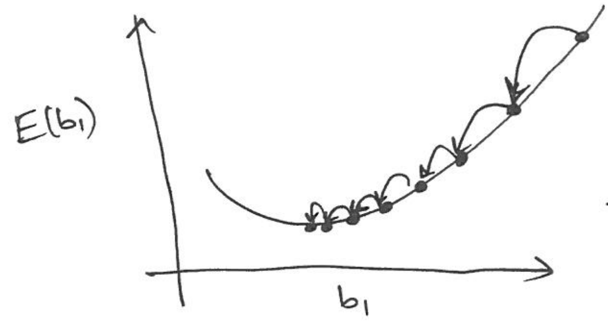
Local Minimum:



slope = 0 at b_1

- $\frac{d}{db_1} E(b_1) = 0.$
- $b_1 := b_1 - \alpha \cdot 0$

  $b_1 := b_1$
- No change in b_1.



- As we approach local minimum, GD will automatically takes smaller step. decrease
- So no need to adjust α value over time.

Apply GD algo in Linear Regression

GD

repeat until convergence

$$\{ \quad b_j := b_j - \alpha \frac{\partial}{\partial b_j} E(b_0, b_1) \quad \text{(for } j=0 \text{ and } j=1) \quad \}$$

L.R (Multivariate)

$$y(x) = b_0 + b_1 x$$

$$E(b_0, b_1) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

$$\underset{(b_0, b_1)}{\text{minimize}} \; E(b_0, b_1)$$

$$\frac{\partial}{\partial b_j} E(b_0, b_1) = \frac{\partial}{\partial b_j} \left(\frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \right)$$

$$= \frac{\partial}{\partial b_j} \frac{1}{2N} \sum_{i=1}^{N} \left((b_0 + b_1 x) - y_i \right)^2$$

$$j=0 \rightarrow \frac{\partial E(b_0, b_1)}{\partial b_0} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)$$

$$j=1 \rightarrow \frac{\partial}{\partial b_1} E(b_0, b_1) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)(x_i)$$

Eq. 1

GD. for LR:

repeat until convergence

$$\{ \quad b_0 := b_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)$$

$$b_1 := b_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)(x_i) \quad \}$$

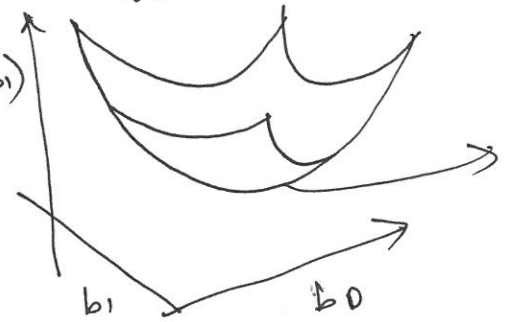
// simultaneous update.

(6)

Note:

- GD suffers from local minimum solutions.
- The $\boxed{\text{Error fn}}$ for Linear Regression is always CONVEX fn.

Sum of Sq. error



convex fn.
- no local minimum.
- only global minimum.

GD for Multivariate Variables (Linear Regression)

- The example we saw is for 1-variable x, we have b_0 and b_1 param to be estimated
- if more than 1-variable, then Eq. 1 does not hold.

$\therefore$ for $X = (x_0, x_1, x_2, \dots x_d)^T$, $\boxed{y = \underbrace{b_0 x_0}_{\text{always } x_0 = 1} + b_1 x_1 + \dots + b_d x_d}$

	x_1	x_2	...	x_d	y
1	x_{11}	x_{12}	...	x_{1d}	y_1
2	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$					
n	x_{n1}	x_{n2}	...	x_{nd}	y_n

} training samples

#n samples.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad B = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_d \end{bmatrix} \qquad X \in R^{d+1} \quad B \in R^{d+1}$$

$$\begin{aligned} y &= b_0 x_0 + b_1 x_1 + \dots + b_d x_d \\ &= B^T \cdot X \\ &= \underbrace{\begin{bmatrix} b_0 & b_1 & \dots & b_d \end{bmatrix}^T}_{(d+1)\times 1 \text{ matrix}} \cdot \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}}_{1 \times (d+1) \text{ matrix}} \end{aligned}$$

$\boxed{y = B^T \cdot X}$ Regression Eq. for Multiple Variables.

⑦

<u>Cost / Error fn:</u>

$$E(B) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

$(b_0, b_1, \dots b_d)$

$\boxed{\begin{array}{l} \text{Repeat until convergence } \{ \\ \qquad b_j := b_j - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{ij} \\ \qquad (\text{Simultaneously update } b_j \text{ for } j = 0, 1, \dots, d) \\ \} \end{array}}$

<u>Example:</u>

$$b_0 := b_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)\, \underbrace{x_{i0}}_{\text{always } 1.}$$

$$b_1 := b_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{i1}$$

$$\vdots$$

$$b_d := b_d - \alpha \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)\, x_{id}$$

<u>GD for Linear Reg</u> (<u>Practical ideas to improve in Real time</u>)
(performance)

1. Make sure features are on a similar scale.

   Eg: x_1 = size (0 — 2000 sq.ft) → need to normalize to 0 to 1. / or standardize
   x_2 = # of rooms (1 – 5)

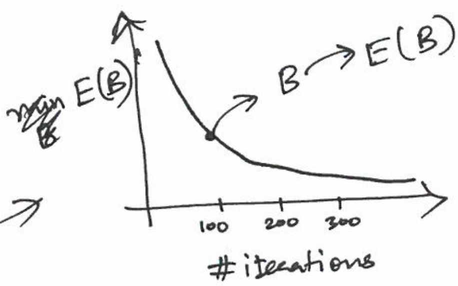
   – Make every features are in range $(-1 \le x_i \le 1)$ roughly.

   $0 \le x_1 \le 3$ ✓

   $-2 \le x_2 \le 0.5$ ✓

   $-100 \le x_3 \le 100$ ✗

   $-0.0001 \le x_4 \le 0.0001$ ✗

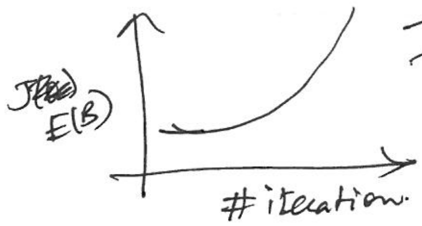


② Plot E(B) graph

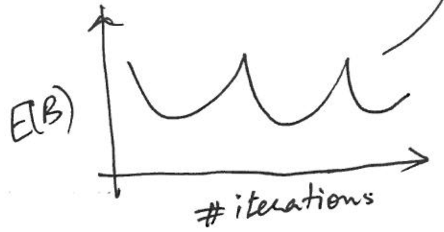
   – E(B) should decrease after every iterations.

8

Gradient Descent - Not Working.



→ check No bugs in code.
→ change α value.
(smaller α value)



- if α is too small, then GD will be very slow to converge.
- if α is too large, E(B) will not decrease for every iterations.
  ↓
  may not converge.

- try α = 0.001, 0.01, 0.1, 1, ... then vary the α-value in this interval
  ↓
  0.003, 0.002