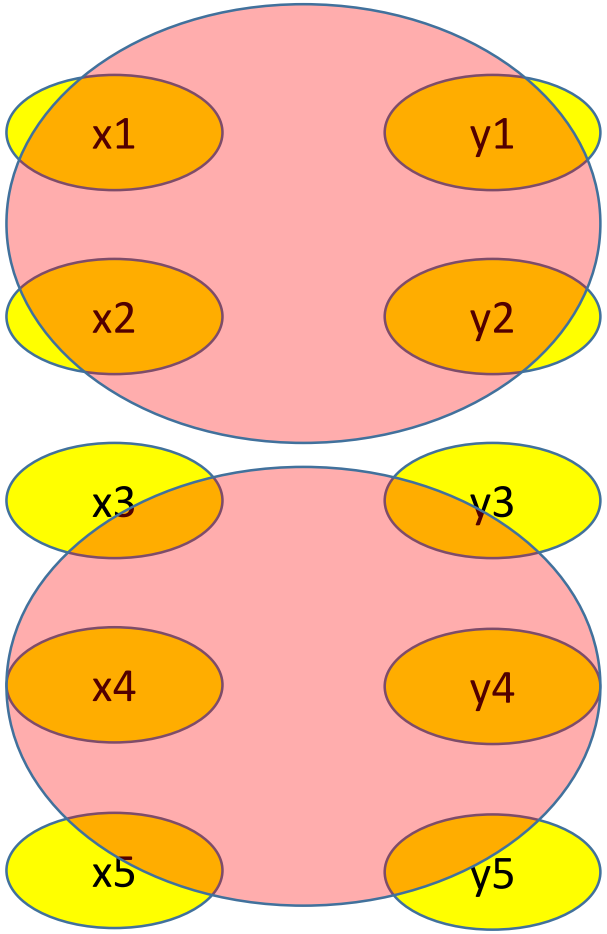
2D Steepest Descent

Dr. Kalidas Y.

In this lecture you will understand how to formulate 2D steepest descent

Linear Regression

Left side (X) → Machine Learning → Right side (Y)



Equation of Line :- $\mathrm{y} = \mathrm{m} * x + c$

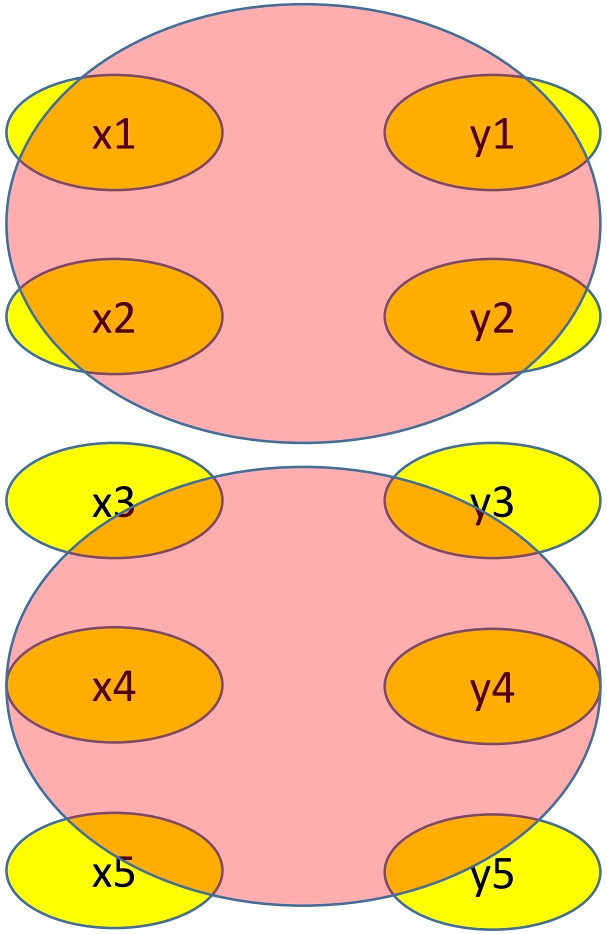
$$\mathrm{L(m, c)} = \sum_{i=1}^{i=5} (y_i - (m * x_i + c))^2$$

$$\frac{\partial L}{\partial m}, \frac{\partial L}{\partial c}, \frac{\partial^2 L}{\partial m^2}, \frac{\partial^2 L}{\partial c^2}, \frac{\partial^2 L}{\partial m \partial c}, \frac{\partial^2 L}{\partial c \partial m}$$

You can treat it like any other ***parameterized*** function

Linear Regression

Left side (X) → Machine Learning → Right side (Y)



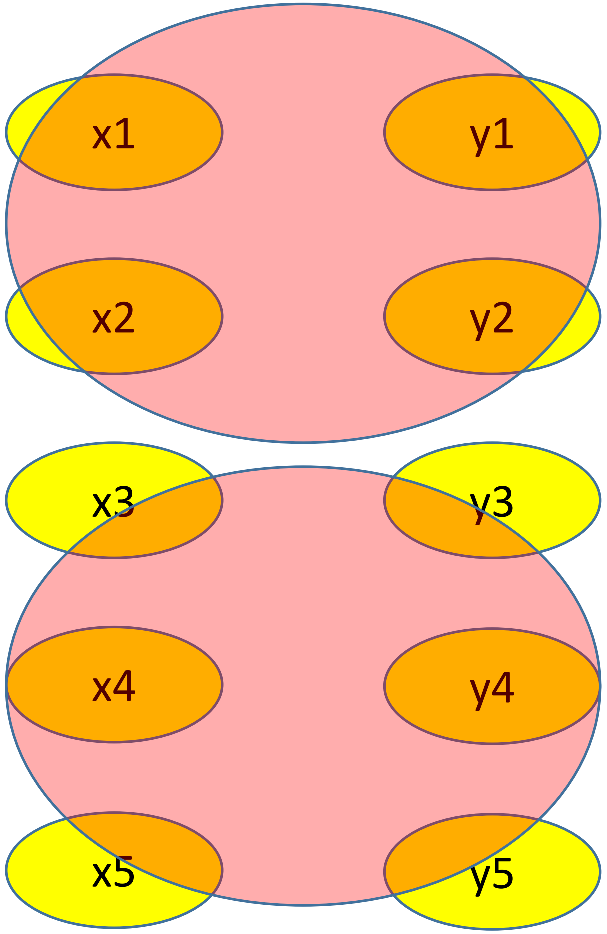
Equation of Line :- $\mathbf{y} = \mathbf{m} * x + c$

$$\mathrm{L(m,c)} = \sum_{i=1}^{i=5} (y_i - (m * x_i + c))^2$$

$$\frac{\partial L}{\partial m} = \sum_{i=1}^{i=5} 2 * (y_i - (m * x_i + c))^1 * (-x_i)$$

Linear Regression

Left side (X) → Machine Learning → Right side (Y)



Equation of Line :- $\mathrm{y} = \mathrm{m} * x + c$

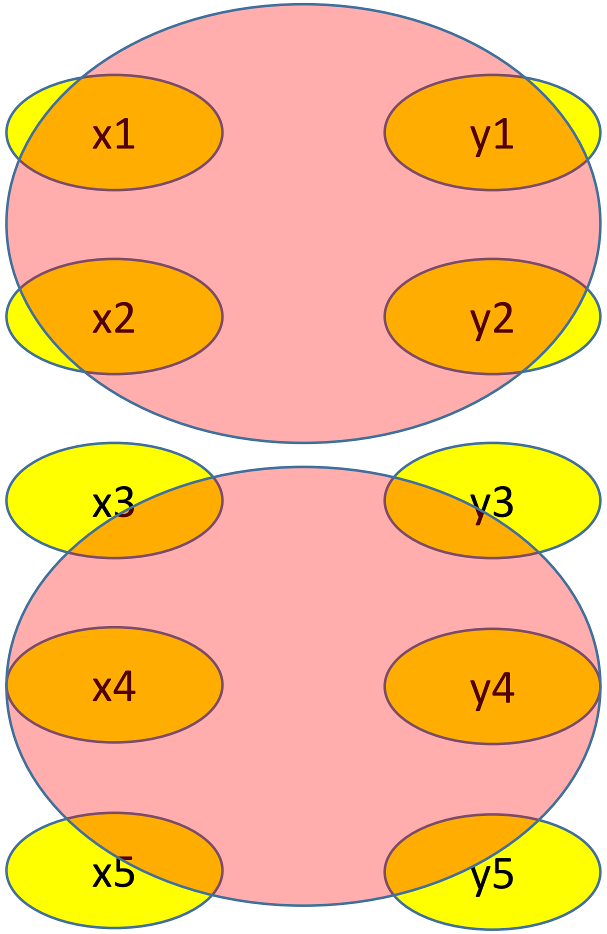
$$\mathrm{L(m, c)} = \sum_{i=1}^{i=5} (y_i - (m * x_i + c))^2$$

$$\frac{\partial L}{\partial m} = \sum_{i=1}^{i=5} 2 * (y_i - (m * x_i + c))^1 * (-x_i)$$

$$\frac{\partial L}{\partial c} = \sum_{i=1}^{i=5} 2 * (y_i - (m * x_i + c))^1 * (-1)$$

Linear Regression

Left side (X) → Machine Learning → Right side (Y)



Equation of Line :- $y = \mathrm{m} * x + c$

$$L(\mathrm{m}, \mathrm{c}) = \sum_{i=1}^{i=5} (y_i - (m * x_i + c))^2$$

$$\frac{\partial L}{\partial m} = \sum_{i=1}^{i=5} 2 * (y_i - (m * x_i + c))^1 * (-x_i)$$

$$\frac{\partial L}{\partial c} = \sum_{i=1}^{i=5} 2 * (y_i - (m * x_i + c))^1 * (-1)$$

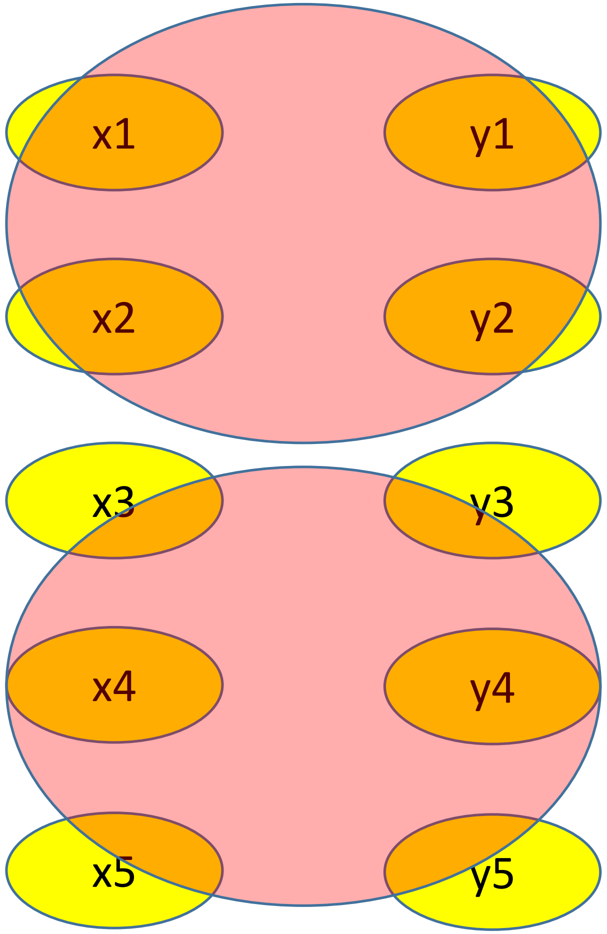
$$\frac{\partial^2 L}{\partial m^2} = \sum_{i=1}^{i=5} 2 * (-x_i) * (-x_i) > 0$$

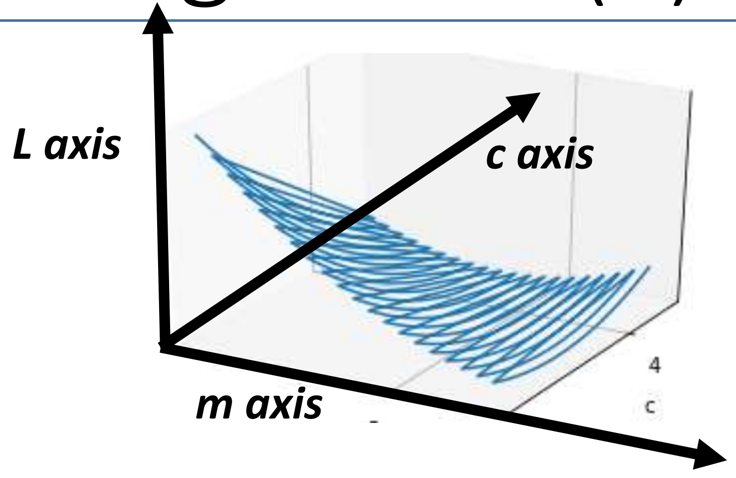
$$\frac{\partial^2 L}{\partial c^2} = \sum_{i=1}^{i=5} 2 * (-1) * (-1) > 0$$

$$\frac{\partial^2 L}{\partial m\, \partial c} = \sum_{i=1}^{i=5} 2 * (-1) * (-x_i) = \frac{\partial^2 L}{\partial c\, \partial m}$$

Linear Regression

Left side (X) → Machine Learning → Right side (Y)





Equation of Line :- $y = m * x + c$

$$L(m, c) = \sum_{i=1}^{i=5} (y_i - (m * x_i + c))^2$$

$$= \sum_{i=1}^{i=5} (y_i^2 + m^2 x_i^2 + 2\, m\, x_i\, c + c^2 - 2\, y_i\, m\, x_i - 2\, y_i\, c)$$

L(m1,c1)
L(m2,c2)
…
…
…
L(mN,cN)

$$= \boldsymbol{\alpha_1 m^2 + \beta_1 m + \alpha_2 c^2 + \beta_2 c + \gamma}$$

*For each given, c, the above curve is a **parabola** in m and L axes*
*For each given m, the above curve is a **parabola** in c and L axes*

…sum of two parabolas NEED NOT BE a parabola

…some kind of a bowl shaped surface

Consider parabola 1: $y = 3\,x^2 + 4$

Consider parabola 2: $y = -\,3\,x^2 + 7x$

parabola 1 + parabola 2: $y = 7x + 4$

This is a line! Not a parabola

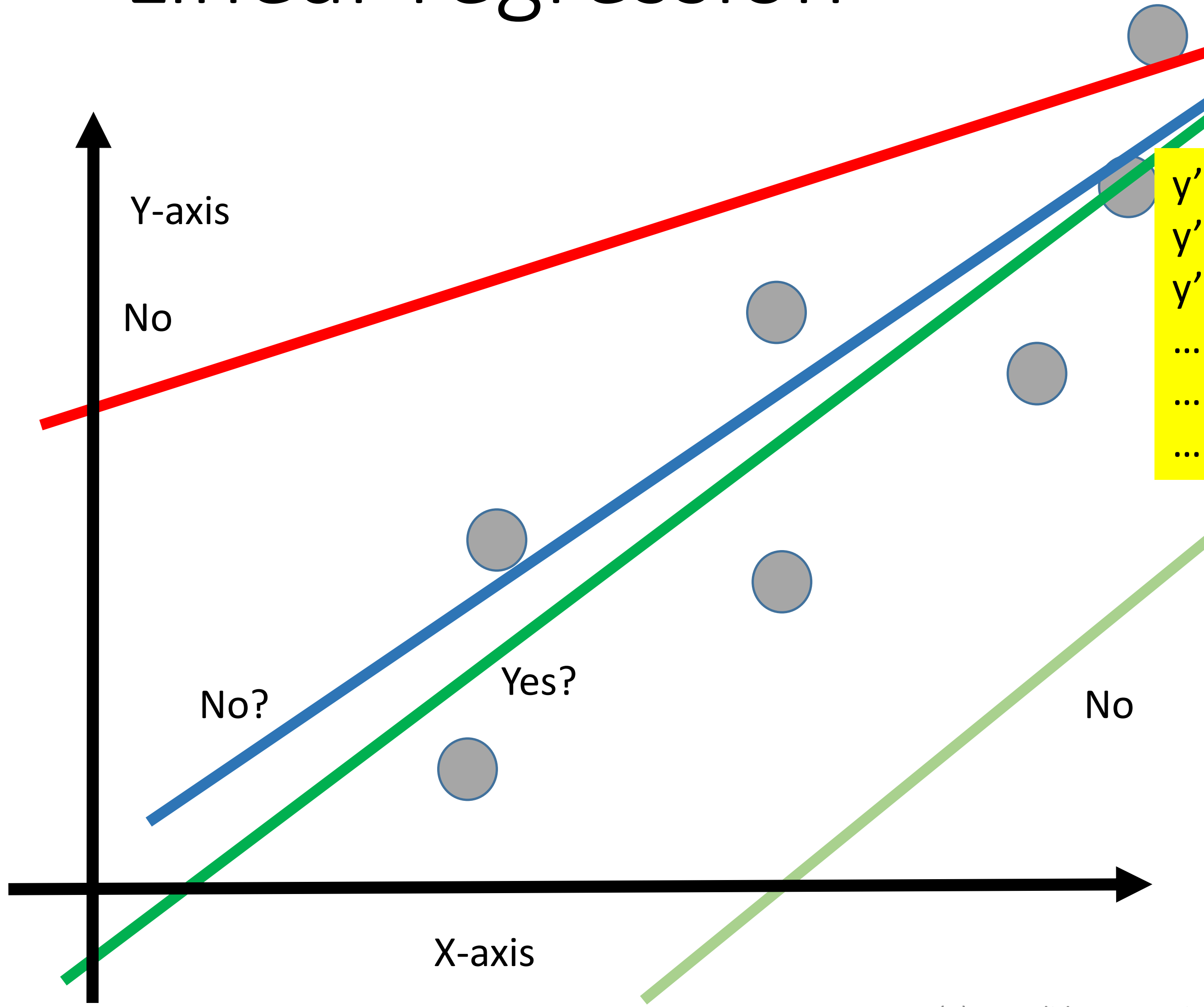
Sum of degree-K polynomials → A polynomial with degree $\le K$

Consider parabola 1: $y = 3\,x^2 + 4$

Consider parabola 2: $y = \sqrt{7}\,x^2 + 8x - 13$

parabola 1 + parabola 2: $y = \left(3 + \sqrt{7}\right)x^2 + 8x - 9$ ***This is a parabola again!***

Linear regression



$(m^1, c^1) \rightarrow (m^2, c^2) \rightarrow \dots \rightarrow (m^n, c^n)$

F(m,c,x) = m x + c

y'1 = m*x1 + c
y'2 = m*x2 + c
y'3 = m*x3 + c
…
…
…

y'1 = m'*x1 + c'
y'2 = m'*x2 + c'
y'3 = m'*x3 + c'
…
…
…

<u>L(m,c)</u> =
$(y1-F(m,c,x1)^2$ +
$(y2-F(m,c,x2)^2$ +
$(y3-F(m,c,x3)^2$ +
…
…
…

Left side

X-coordinate Object
- x coordinate value

vector of 1 element

Right side

Y-coordinate Object
- y coordinate value

vector of 1 element

Matrix Formulation of L(m,c)

- $X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \end{bmatrix}, W = \begin{bmatrix} m \\ c \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}$

- $L(W) = (XW - Y)^T (XW - Y)$

Gradient Descent Formulation

Loss function of 2D vector

- $L(W) = (XW - Y)^T(XW - Y)$
- $L(W) = \left(\begin{bmatrix} x_1 & 1 \\ \multicolumn{2}{c}{\cdots} \\ x_n & 1 \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}\right)^T \left(\begin{bmatrix} x_1 & 1 \\ \multicolumn{2}{c}{\cdots} \\ x_n & 1 \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}\right)$

- $L(W) = \sum_{i=1}^{i=n}\left(x_{i,1}w_1 + 1 * w_2 - y_i\right)^2$

Gradient of Loss function (2D scenario)

- $L(W) = \sum_{i=1}^{i=n}\left(x_{i,1}w_1 + w_2 - y_i\right)^2$

column 1

- $\frac{\partial L(W)}{\partial w_1} = \sum_{i=1}^{i=n} 2 * \left(x_{i,1}w_1 + w_2 - y_i\right)^1 x_{i,1} \; = 2 * X[:,1]^T(XW - Y)$
- $\frac{\partial L(W)}{\partial w_2} = \sum_{i=1}^{i=n} 2 * \left(x_{i,1}w_1 + w_2 - y_i\right)^1 = 2 * X[:,2]^T(XW - Y)$

column 2

- $\nabla L(W) = \begin{bmatrix} \frac{\partial L(W)}{\partial w_1} \\ \frac{\partial L(W)}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 2 * X[:,1]^T(XW - Y) \\ 2 * X[:,2]^T(XW - Y) \end{bmatrix} = 2 * \begin{bmatrix} X[:,1]^T \\ X[:,2]^T \end{bmatrix} * (XW - Y)$
- $= 2\, X^T(XW - Y)$

Alternatively…

- Start, $m = c = 0$ *//or any random*
- Iterate 1000 steps (*or whatever*)
  - $m_{(new)} = m_{(old)} - \eta \frac{\partial L}{\partial m}|_{m=m_{(old)},c=c_{(old)}}$
  - $c_{(new)} = c_{(old)} - \eta \frac{\partial L}{\partial c}|_{c=c_{(old)},m=m_{(new)}}$
- Output m, c

Steepest Descent for 2D Loss function

This is a ***step size***

a.k.a ***learning rate***

$$\boldsymbol{W_{(new)} = W_{(old)} - \eta \nabla L(W) \Big|_{W=W_{(old)}}}$$

This is a ***function***

Function computed ***at***

specified point

of interest

What are the different ways?

2nd argument is
the list of Parameters

BruteForceSolver(L, [m1,c1])

1st argument is the Loss function

$\boldsymbol{minval = +10000.0}$

FOR $\mathbf{m} \in [\mathbf{-100, \dots, +100}]$

FOR $\mathbf{c} \in [\mathbf{-100, \dots, +100}]$

Compute $\boldsymbol{v = L(m, c)}$

IF $\boldsymbol{v < minval}$:
$\boldsymbol{v = minval}$
$\boldsymbol{m1 = m,}$
$\boldsymbol{c1 = c}$

Questions…

- *-100 to +100, who gave the range?*
- *What is the step size?*
- *What if the solution is highly fine, (3.451, -89.1123)*
- *Can we speed up?*

What are the different ways?

2nd argument is
the list of Parameters

RandomSolver(L, [m1,c1])

1st argument is the Loss function

$minval = +10000.0$

FOR ITER= 1:100000

FOR $\mathbf{m} = RAND(-100, 100)$

FOR $c = RAND(-100, 100)$

Compute $v = L(m, c)$

IF $v < minval$:
$v = minval$
$m1 = m,$
$c1 = c$

Questions…

- *-100 to +100, who gave the range?*
- *What is the step size?*
- *What if the solution is highly fine, (3.451, -89.1123)*
- *Can we speed up?*

What are the different ways?

2nd argument is *the list of Parameters*

```
GradiantSolver( L, [m1,c1], gradL)
```

? ? ? ? ?

1st argument is the Loss function

3rd argument is the gradient function

What are the different ways?

2nd argument is
the list of Parameters

GradiantSolver(L, [m,c])

1st argument is the Loss function

- **INIT m,c = 0**
- **Iterate 1000 steps** (*or whatever*)
  - $\boldsymbol{m_{(new)} = m_{(old)} - \frac{\partial L}{\partial m}\Big|_{m=m_{(old)}}}$
  - $\boldsymbol{c_{(new)} = c_{(old)} - \frac{\partial L}{\partial c}\Big|_{c=c_{(old)}}}$

What are the different ways?

2nd argument is
the list of Parameters

SteepestDescentSolver(L, [m1,c1], gradL)

//m1,c1 = some initial value is already input to the function

- **Iterate 1000 steps (*or whatever*)**
- **IF L(m1,c1) < Threshold**
  - **RETURN**
- **[m1,c1] = [m1,c1] – gradL(m1,c1)**

This function outputs a list or vector

1st argument is the Loss function

3rd argument is
the gradient function

Result being used: A symmetric matrix is positive semidefinite if and only if all its eigenvalues are nonnegative.

To prove: The matrix $G = \begin{bmatrix} z & \overline{x} \\ \overline{x} & 1 \end{bmatrix}$ is positive semidefinite.

Proof. Since G is a symmetric matrix, by the above result, it is enough to prove that all the eigenvalues of the matrix G are nonnegative. The eigenvalues of the matrix G are given by the roots of the equation $(z-\lambda)(1-\lambda) - \overline{x}^2 = 0$.
That is, the roots of the equation

$$\lambda^2 - (z+1)\lambda + z - \overline{x}^2 = 0. \tag{1}$$

This gives
$$\lambda = \frac{(z+1) \pm \sqrt{(z+1)^2 - 4(z-\overline{x}^2)}}{2}.$$

Claim 1. All the roots of equation (1) are real numbers.
Proof of Claim 1. The discriminant of equation (1) is given by $(z+1)^2 - 4(z-\overline{x}^2)$.
Now,

$$\begin{aligned}(z+1)^2 - 4(z-\overline{x}^2) &= z^2 + 2z + 1 - 4z + 4\overline{x}^2 \\ &= z^2 - 2z + 1 + 4\overline{x}^2 \\ &= (z-1)^2 + 4\overline{x}^2,\end{aligned}$$

which is nonnegative, as it the sum of squares of real numbers.
This proves Claim 1.

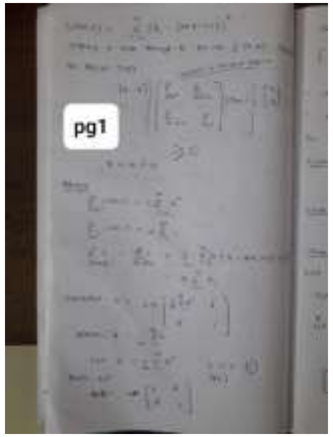
Claim 2. All the roots of equation (1) are nonnegative.
Proof of Claim 2. By Jensen's inequality, we have $\overline{x}^2 \le z$. This yields

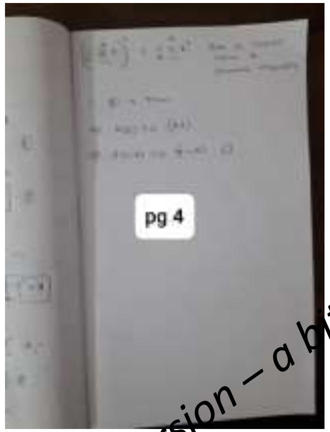
$$\begin{aligned} \overline{x}^2 &\le z \\ \Rightarrow -4(z-\overline{x}^2) &\le 0 \\ \Rightarrow (z+1)^2 - 4(z-\overline{x}^2) &\le (z+1)^2 \\ \Rightarrow \sqrt{(z+1)^2 - 4(z-\overline{x}^2)} &\le z+1 \quad (\text{Since } (z+1) > 0) \\ \Rightarrow \frac{(z+1) \pm \sqrt{(z+1)^2 - 4(z-\overline{x}^2)}}{2} &\ge 0 \end{aligned}$$

That is, the roots of the equation (1), and hence the eigenvalues of the matrix G are nonnegative. Hence the matrix G is positive semidefinite.

Remarks.

1. Here, Claim 1 (nonnegativity of the discrminant) is redundant, as the proof of Claim 2 does not assume that the roots are real.
2. The quadratic loss function is infinitely differentiable. So the Hessian matrix is symmetric and the above technique can be used to check if the Hessian is positive semidefinite in the multivariate scenario, with even more than two variables.

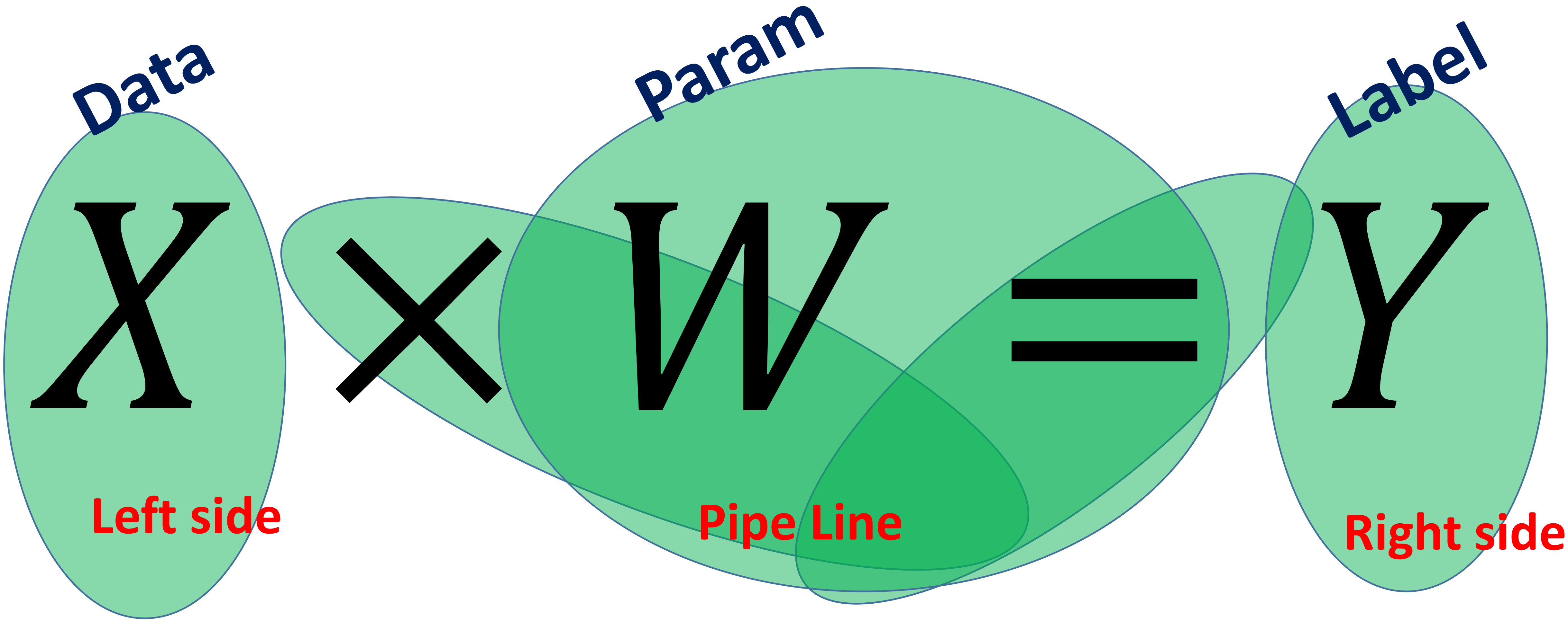




Dr. Kalidas's version – a bit lengthier proof though

Thanks to Dr. Lakshmi, IISER Mathematics Faculty for the proof on left (simplified version)

Data, Label, Parameter – Simple supervised case



$$X \times W = Y$$