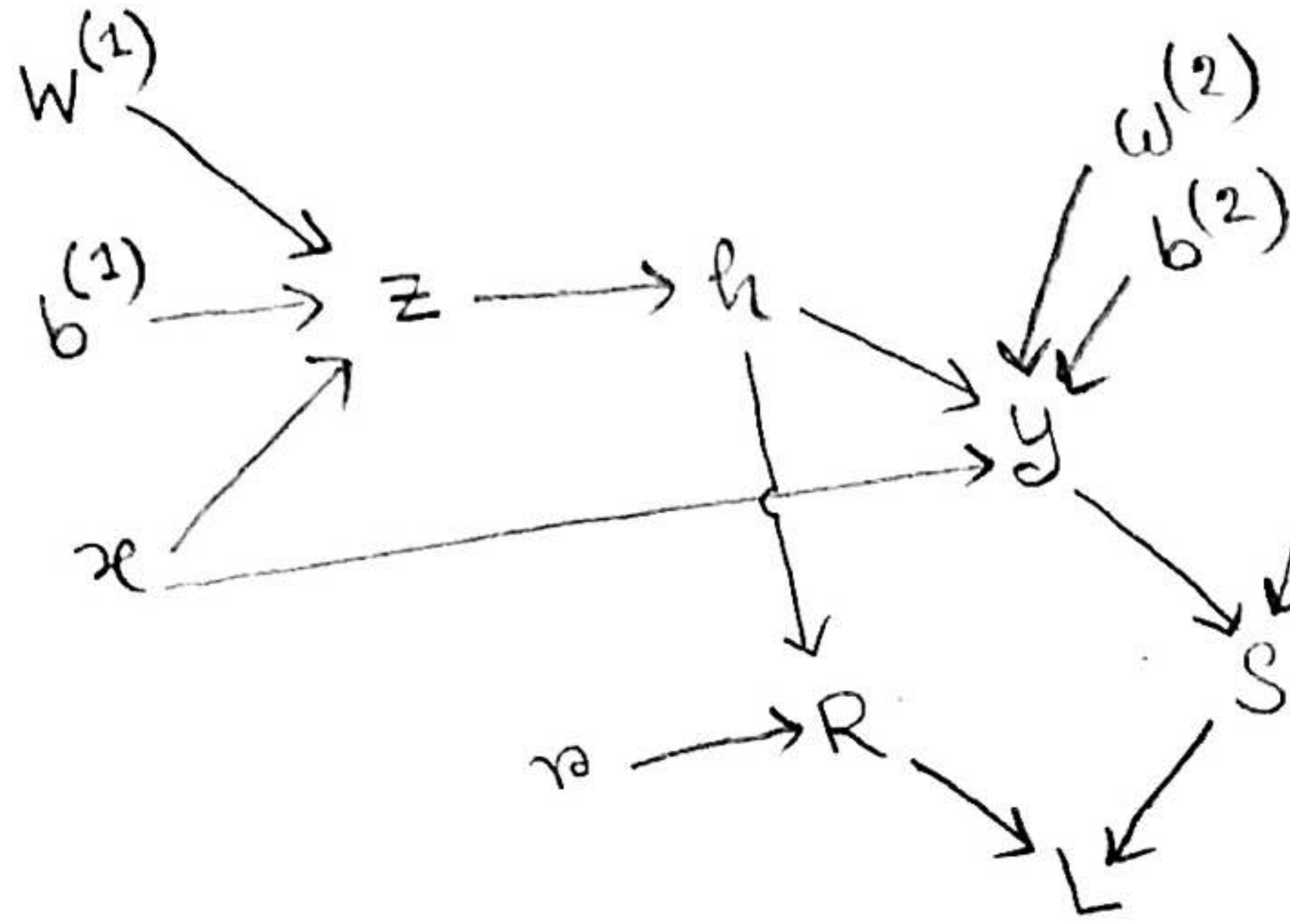
Asadullah Hill Galib

CSE-891 : Homework 2

October, 11

1 1



1 2 $\frac{dL}{dx} = x' = ?$

$L' = 1$

$S' = L'$

$R' = L'$

$y' = S'(y - s)$

$h' = y' . w^{(2)} + R' . r^T$

$z' = h' . \sigma'(z)$

$x' = z' w^{(1)} + y' . 1$

(Ans)

(2) (1)

$f(x) = vv^T x$

$J = vv^T$

$n = 3, \ v^T = [1, 2, 3]$

$$J = vv^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1 \ 2 \ 3] = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

(2) (2)

The time and memory cost of evaluating the Jacobian is $O(n^2)$. [There are $n \times n$ calculation & $n \times n$ grids ~~in~~ ~~or~~ in memory are needed]

(2) (3)

$z = J^T y$

$= vv^T y$ [transpose of $J = J$]

Here, instead of evaluating $(v.v^T).y$, one should do this : $v.(v^T.y)$

$$\underbrace{v.\underbrace{(v^T.y)}_{\text{first} \rightarrow \text{linear in } n \ [1 \times 1]}}_{\text{second} \rightarrow \text{linear in } n \ [3 \times 1]}$$

So, $z = J^T y$

~~So~~, ~~z~~ $v.(v^T y)$

$$= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [1 \cdot 2 \ 3] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot 6 = \begin{bmatrix} 6 \\ 12 \\ 18 \end{bmatrix} \text{(Ans)}$$

(3) (1)

Loss, $\mathcal{L} = \frac{1}{n} \| X\hat{\omega} - t \|_2^2$

$$\frac{d\mathcal{L}}{d\hat{\omega}} = \cancel{\frac{2}{n}}\cancel{\|(X\omega} \; \frac{2}{n} X^T (X\hat{\omega} - t)$$

$$= \frac{2}{n} \left(X^T X \hat{\omega} - X^T t \right)$$

(3) (2) Underparameterized:

(a)

$$\frac{d\mathcal{L}}{d\hat{\omega}} = 0$$

$$\Rightarrow \frac{2}{n} (X^T X \hat{\omega} - X^T t) = 0$$

$$\Rightarrow X^T X \hat{\omega} = X^T t$$

$$\therefore \hat{\omega} = (X^T X)^{-1} X^T t$$ [as $X^T X$ is invertible for $n > d$]

(3) (2) (b) $t_i = \omega^{*T} x_i$.

$$\cancel{t = X w^*} \quad t = X\omega^*$$

So, $\hat{\omega} = (X^T X)^{-1} X^T X \omega^* = \omega^*$

Therefore, $\forall x \in \mathbb{R}^d$, $(\omega^{*T} x - \hat{\omega}^T x)^2 = 0$ [when $d < n$]

and $\hat{\omega}$ achieves perfect generalization.

③ ③ Overparameterized Model: 2D

ⓐ

$n = 1,\ d = 2,\ x_1 = [2\ \ 1] \quad y_1 = 2$

$\hat{\omega}^T x_1 = y_1$. Let, $\hat{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}$

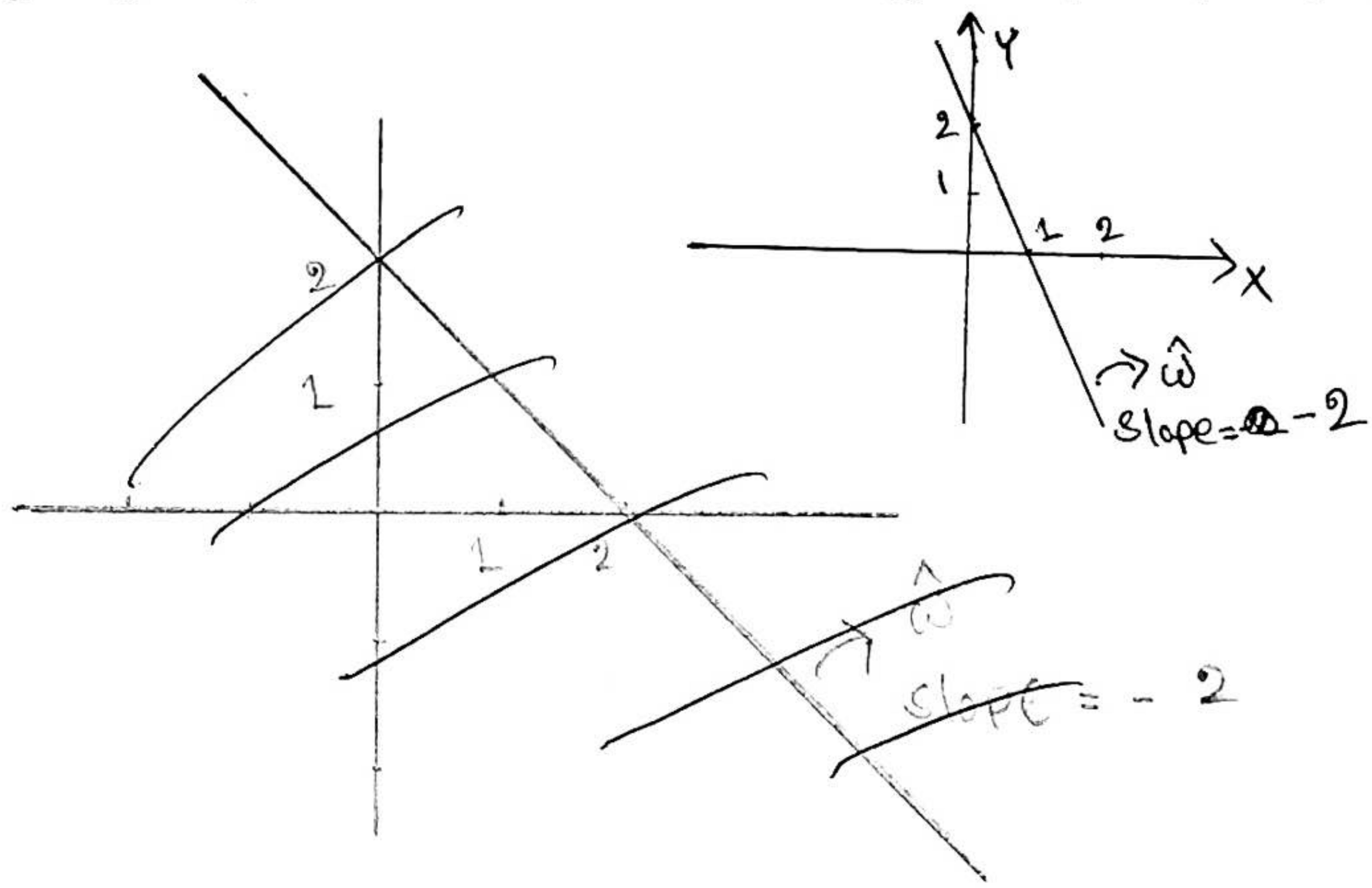
So, $[\omega_1\ \ \omega_2]\begin{bmatrix} 2 \\ 1 \end{bmatrix} = y_1 = 2$

$\Rightarrow 2\omega_1 + \omega_2 = 2$

$\Rightarrow \omega_2 = -2\omega_1 + 2$ → equation of line

So, there exists infinitely many $\hat{\omega}$, satisfying $\hat{\omega}^T x_1 = y_1$

as ~~there's no~~ the solution can be anywhere on the line.



3 3 b

$\hat{\omega}(0) = 0$

$$\frac{dL}{d\hat{\omega}} = \frac{2}{n} X^T (X\hat{\omega} - t)$$

when, $\hat{\omega}(0) = 0$ and ~~$X = X_1$,~~ $X = X_1$, $t = t_1$, $n = 1$, $d = 2$:

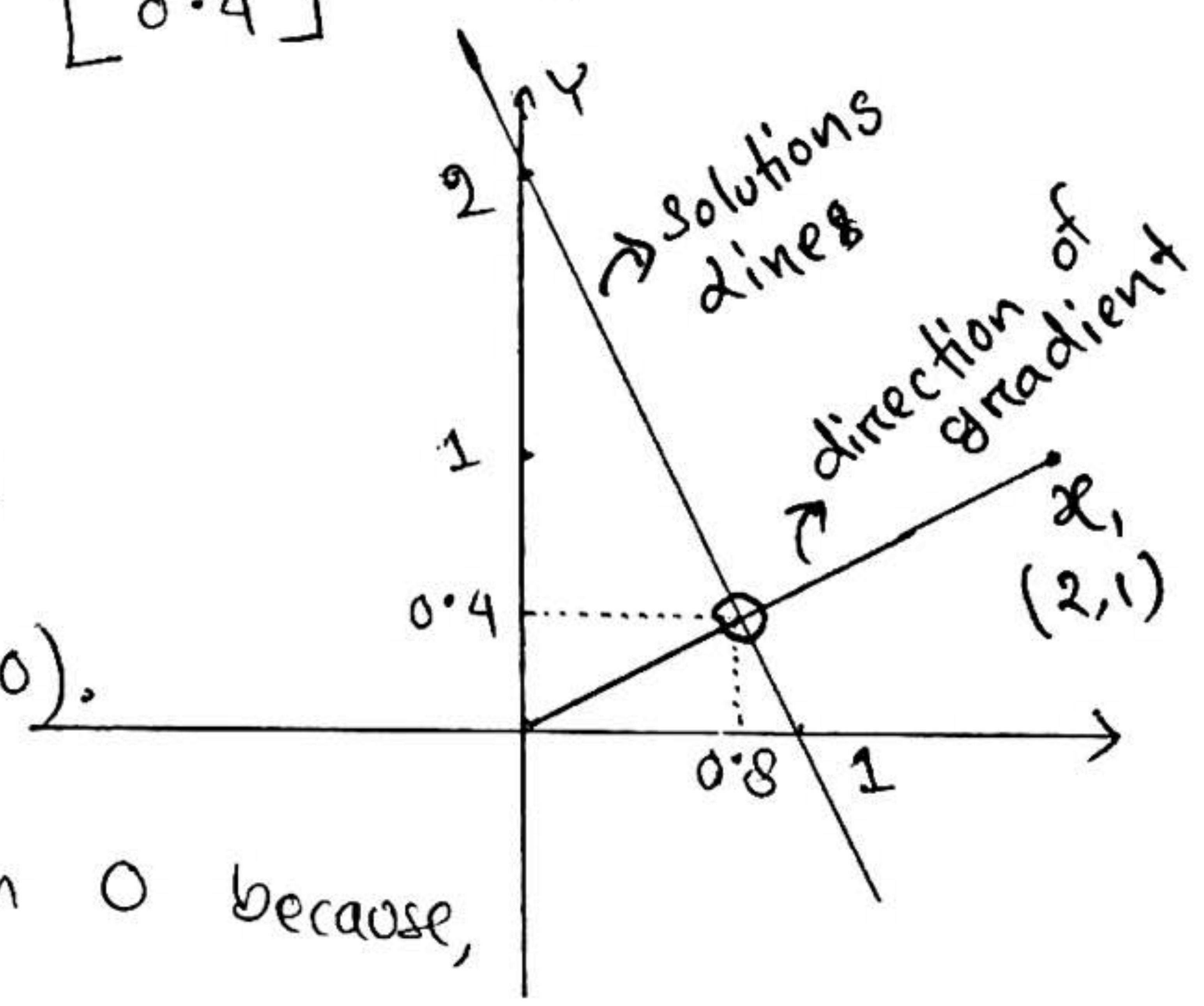
$$\frac{dL}{d\hat{\omega}} = \frac{2}{n}\left(-X^T t\right) = (-2 X_1^T t_1)$$

As, $t_1 = 2$ a constant, the direction of the gradient is along X_1. And, it doesn't change along the trajectory as there is no $\hat{\omega}$ term in the derivative.

The corresponding unit norm vector of X_1 : $\frac{1}{\sqrt{5}}\begin{bmatrix} 2 \\ 1 \end{bmatrix}$

Using the above, we get

$$\hat{\omega} = 2 \cdot \frac{1}{(\sqrt{5})^2}\begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix}$$ (using squared-norm)



Here,

the gradient descent finds the closest solution from w_0 (0,0).

It is the closest distance from 0 because, the direction of gradient and solutions line are orthogonal to each other (i.e. slope of the solutions line, $m_1 = -2$

" " " gradient des. ", $m_2 = \frac{1}{2}$

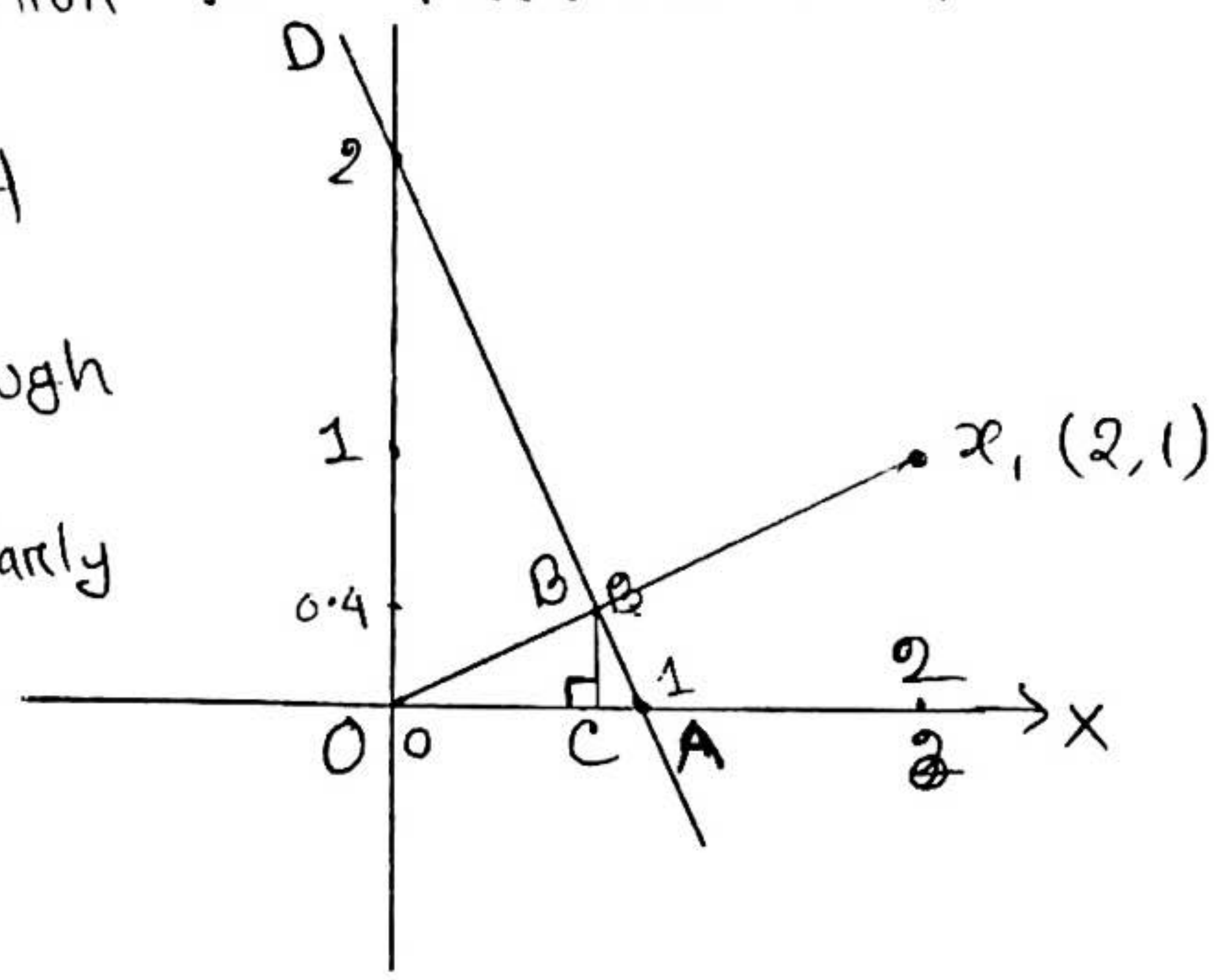
$m_1 m_2 = -1 \Rightarrow$ perpendicular)

③③ⓒ

Using Pythagoream Theorem:

B is the gradient descent solution from previous part.

To show, B has the smallest Euclidean norm, it is enough to prove that OB perpendecularly intersects AD ($OB \perp AD$)



Here, $OA = 1$

$OC = 0.8$

$BC = 0.4$ [$BC \perp OA$]

So, $OB = \sqrt{OC^2 + BC^2} = \sqrt{\frac{2}{5}}$

$$AB = \sqrt{CA^2 + BC^2} = \sqrt{(OA - OC)^2 + BC^2}$$

$$= \sqrt{(0.2)^2 + (0.4)^2} = \sqrt{\frac{1}{5}}$$

In ~~ΔOAB~~ ΔOBA,

$$OB^2 + AB^2 = \frac{4}{5} + \frac{1}{5} = 1 = OA^2$$

So, ~~ΔOAB~~ ΔOBA is a right-angled Δ. $\Rightarrow OB \perp AB$

$\Rightarrow OB \perp AD$

[Proved]

—o—

Alternative proof:

⊛ It can also be shown ~~also~~ using slopes:

$$m_{AD} = -2 \quad \text{and} \quad m_{OB} = 1/2$$

So, $m_{AD} \cdot m_{OB} = -1 \Rightarrow OB \perp AD \Rightarrow OB$ has the smallest norm.

(3) (4) (a) With, $\hat{\omega}(0) = 0$, we get gradient vector in the span of X. [previous part suggests]

As the gradient vector is always spanned by the rows of X, we can get $\hat{\omega}$ as a linear combination of X and some other matrix. Let, P is that matrix.

So, $\hat{\omega} = X^T P$

Then, $\frac{2}{n} X^T (X\hat{\omega} - t) = 0$

$$\Rightarrow X^T(XX^TP - t) = 0$$

$$\Rightarrow XX^T(XX^TP - t) = X.0 = 0$$

$$\Rightarrow XX^TP - t = (XX^T)^{-1}.0 = 0$$ [for $d > n$, XX^T is invertible]

$$\Rightarrow XX^TP = t$$

$$\Rightarrow P = (XX^T)^{-1} t$$

Thereby, $\hat{\omega} = X^T P$

$$= X^T (XX^T)^{-1} t$$

The solution is unique as it's just a linear transformation of t.

3 4 b

zero-loss solution with $\hat{\omega}_1$.

So, $\hat{\omega}_1^T x - t = 0 \Rightarrow \hat{\omega}_1^T x = t \; \ldots (1)$

$$(\hat{\omega} - \hat{\omega}_1)^T \hat{\omega} = (x^T (xx^T)^{-1} t - \hat{\omega}_1)^T \hat{\omega}$$

$$= (t^T (xx^T)^{-1} x - \hat{\omega}_1) \hat{\omega} (x^T (xx^T)^{-1} t)$$

$$= (t^T (xx^T)^{-1} x x^T (xx^T)^{-1} t - \hat{\omega}_1 x^T (xx^T)^{-1} t)$$

$$= t^T (xx^T)^{-1} t - t^T (xx^T)^{-1} t \quad [\text{using } (1)]$$

$$= 0$$

So, $(\hat{\omega} - \hat{\omega}_1)$ and $\hat{\omega}$ are perpendicular to each other

$\Rightarrow$ $\hat{\omega}_1$ and $\hat{\omega}$ are perpendicular to each other.

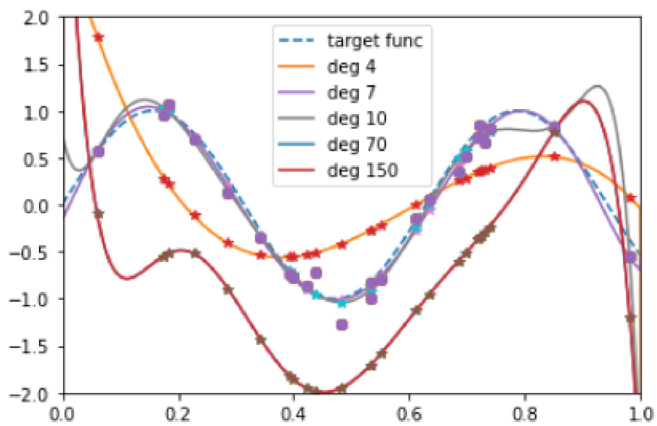
~~To~~ So, like $\hat{\omega}_1$, all other solutions are perpendicular to $\hat{\omega}$. And, this gradient descent solution, $\hat{\omega}$ has the smallest Euclidean norm similarwise we proved before (using pythagream theorem).

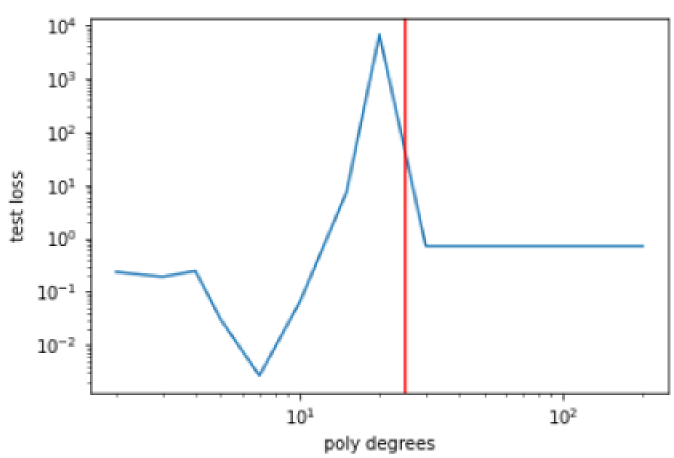
Question 3 -5:

```
# to be implemented; fill in the derived solution for the unc

def fit_poly(X, d, t):
    X_expand = poly_expand(X, d=d, poly_type=poly_type)
    if d > n:
        W = X_expand.T@np.linalg.inv(X_expand@X_expand.T)@t
    else:
        W = np.linalg.inv(X_expand.T@X_expand)@X_expand.T@t
    return W
```

```
2 0.23473638175555447
3 0.19020505096352716
4 0.24537346900180165
5 0.02963124207539845
7 0.002650135078807432
10 0.06635344938407685
15 7.3835492062768155
20 6706.8570800068255
30 0.7170381150111599
50 0.7170381150111599
70 0.7170381150111599
100 0.7170381150111599
150 0.7170381150111599
200 0.7170381150111599
```





No, overparameterization does not always lead to overfitting. Here, overparameterization give stable and better performance than the medium range of parameters (9-35). Implicit regularization induced by gradient descent is reason for this trend.

4 1 SGD:

$$X\hat{\omega} = t \Rightarrow X\hat{\omega} - t = 0 \quad \cdots 1$$

In SGD, all x_i is contained in the span o X. And the SGD update ~~do~~ steps don't ever leave the span of X. Because, $\frac{d}{d\hat{\omega}_p}(x_i\hat{\omega}_p - t_i)^2 = 0$ will ~~giv~~ update $\hat{\omega}_p$ as ~~so~~ some combination of x_i, and t_i.

Thereby, we can assume the SGD solution is spanned by X : $\hat{\omega} = X^T S$, where S is a arbitrary matrix.

From 1, $X\hat{\omega} - t = 0$

$$\Rightarrow XX^T S - t = 0$$

$$\Rightarrow S = (XX^T)^{-1} t$$

So, $\hat{\omega} = X^T(XX^T)^{-1} t$

$$= \omega^*$$

[Showed]

(3)

(4) (2) Mini-batch SGD:

Yes, mini-batch SGD also obtains minimum norm solution on convergence.

Because the batch B is taken from the rows of X.

So, the solution $\hat{\omega}$ is spanned by the rows of X.

$$\hat{\omega} = \underbrace{B}_{\text{Batch}} S = X\tilde{S}$$

So, $X\hat{\omega} - t = XX^T\tilde{S} - t = 0$

$$\Rightarrow \tilde{S} = (XX^T)^{-1} t$$

$$\therefore \hat{\omega} = X^T(XX^T)^{-1} t = \omega^*$$

(4) (3) Adaptive Methods: Adagrad

$x_1 = [2, 1] \quad \omega_0 = [0, 0] \quad t = [2]$

Using minimum norm solution with GD,

we got $\omega^* = \begin{bmatrix} 0.8 \\ 0.4 \end{bmatrix}$ and

$$\nabla_{\omega^*} \mathcal{L}(\omega) = -2 x_1 t_1$$

Using Adagrad,

$$\omega_0 = [0, 0]$$

$$\omega_1 = \omega_0 - \frac{\eta}{\sqrt{G_{\cancel{0}1}} + \epsilon} \nabla_{\hat{\omega}_0} \mathcal{L}(\omega)$$

$$G_1 = 0 + \left(\nabla_{\hat{\omega}_0} \mathcal{L}(\omega)\right)^2$$

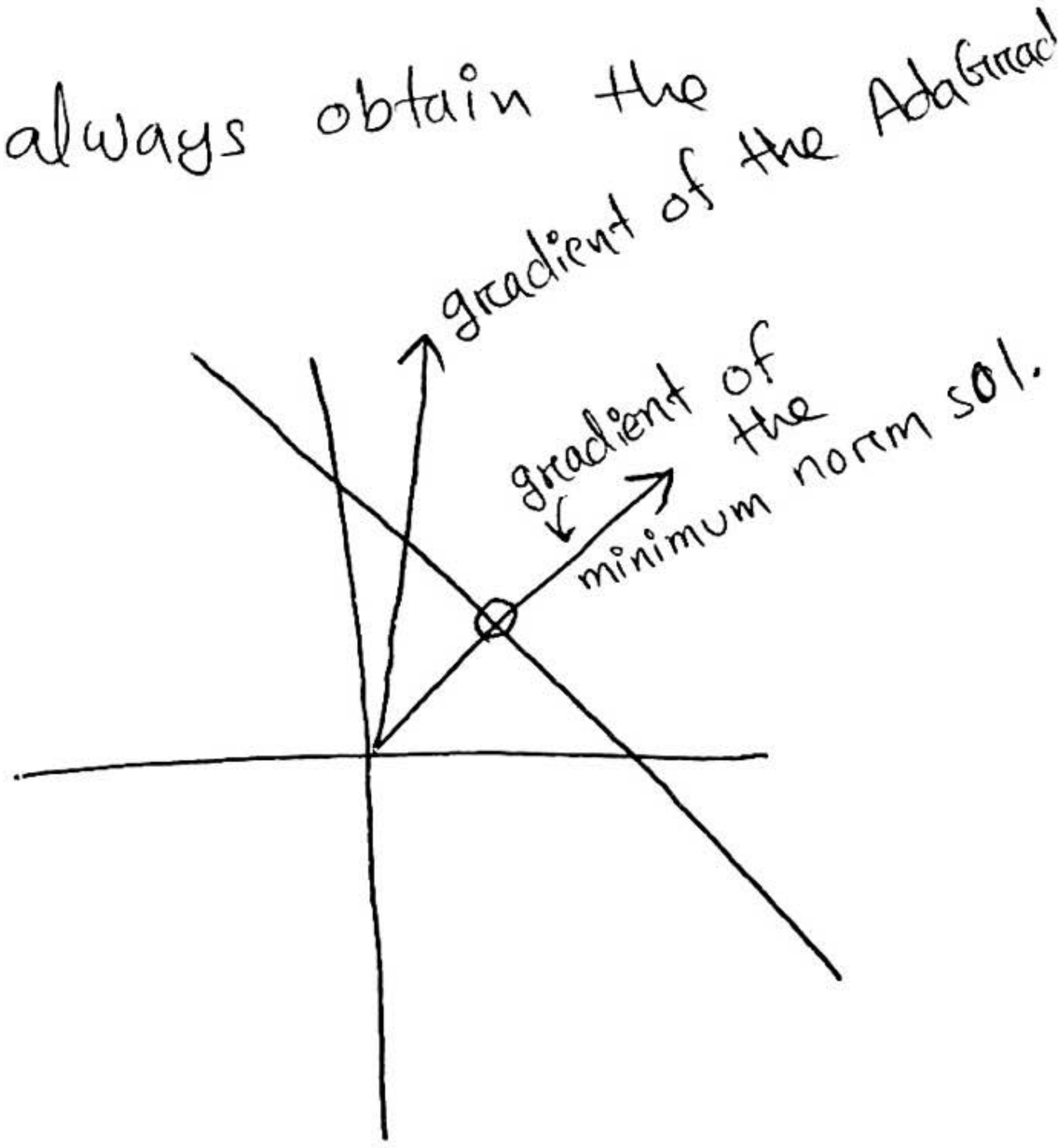
Let, assume, $\nabla_{\hat{\omega}_0} \mathcal{L}(\omega) = -2 x_1 t_1$ [similar to the GD]

then, $\omega_1 = \omega_0 - \frac{\eta}{(-2x_1 t_1) + \epsilon} \cdot (-2x_1 t_1)$

as ϵ is small, ω_1 looses x_1 term almost, ~~that means the direct~~ because numerator and denominator both contains $(-2x_1 t_1)$. So,

w_1 has little impact from x_1, which indicates the direction of the gradient is no longer along x_1 as much as the w^* case.
(GD with minimum norm sol.)

Thereby, AdaGrad doesn't always obtain the minimum norm solution.



This ~~Same~~ results holds true for other adaptive ~~models~~ methods (RMSProp, Adam) in general.

Because the scaling part in the weight update may divert solution gradient from the span of X and it may get outside of the span of X sometimes.