

EECS 245, Winter 2026

LEC 19 Gradient Descent and Convexity

→ Read: Ch. 8

Agenda

- Gradient descent: what's the point?
 - Update rule
 - Worked example
- Using GD to find optimal model parameters
- When is GD guaranteed to converge?
 - Convexity (not on MT2, but on Final Exam)

Announcements

- MT 2 on Monday; see logistics on Ed
- Mock exam Friday: 2:30-5:30 PM, 1014 DOW
- HW 9A due Friday, HW 9B out soon
- No in-person lecture next week:
 - office hours Monday
 - Video posted for Weds.

lots of new visuals
in Ch. 7
and Ch-8:
take a look!

```
.../Desktop/245/notes
.../Desktop/245/notes #1  codex #2  jbs #3  codex #4  +

references now, then I'll wrap with the minimal outcome and verification details.

• Explored
  └─ Read new_grad_utils.py, gd_utils.py

• The GD path titles now include step size everywhere they're built.

I fixed the shared title plumbing in 08_gradients/new_grad_utils.py:450, and passed step_size
through for the contour, slider, and surface figures in 08_gradients/new_grad_utils.py:459 and
08_gradients/new_grad_utils.py:589. The older copies in 08_gradients/gd_utils.py:304 were
updated too, and the surface plot there now actually applies its title.

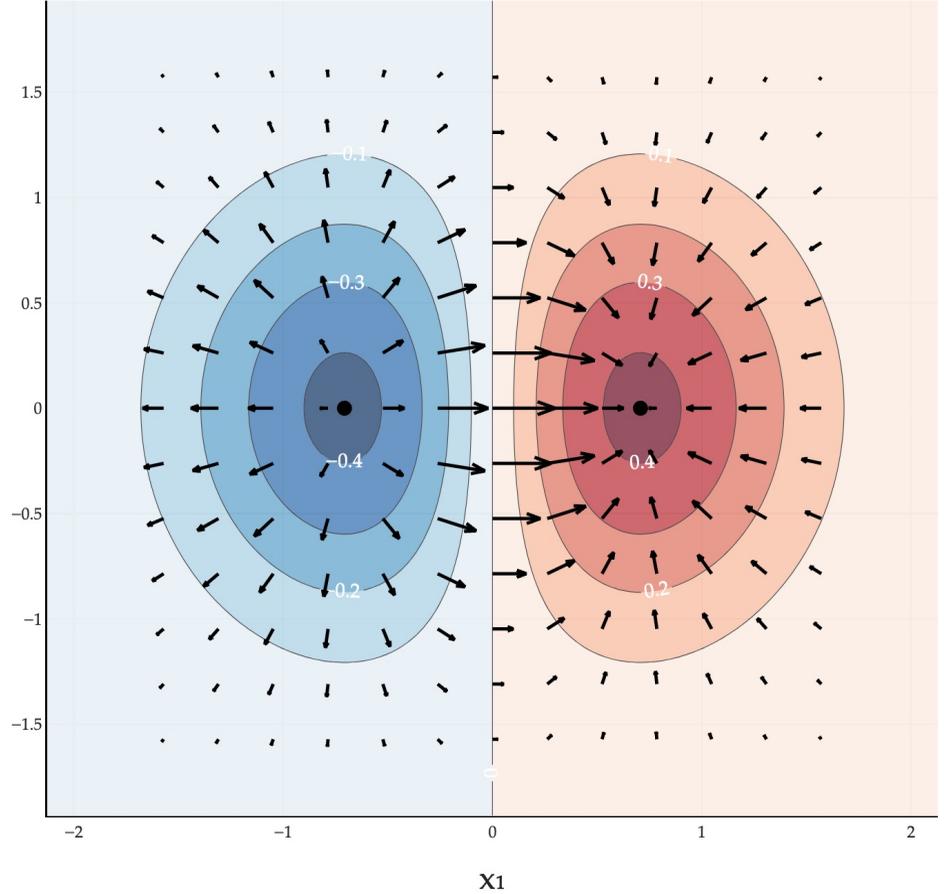
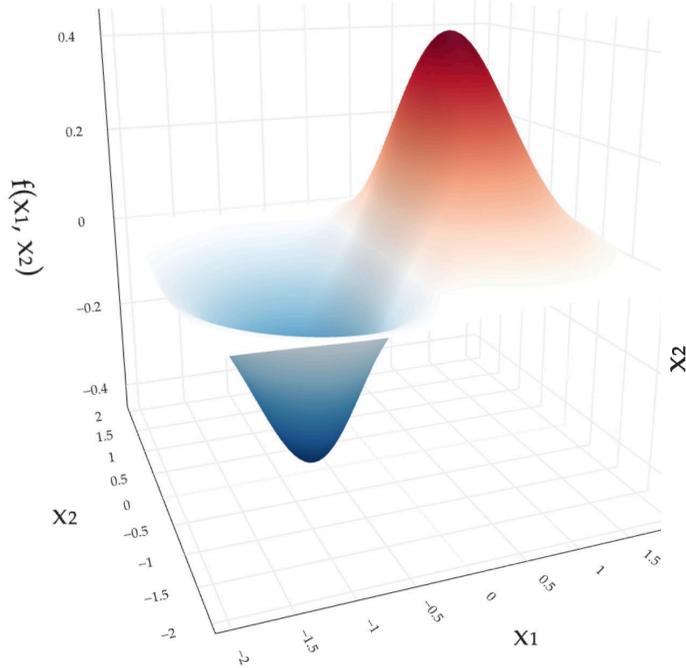
I also refreshed the stored Plotly outputs in 08_gradients/03-gradient-descent.ipynb, so the
notebook now shows:

- static contour: ..., Step Size = 0.1
- static contour: ..., Step Size = 0.25
- surface: ..., Step Size = 0.25
- slider: ..., Step Size = 0.05 plus the iteration label on every slider step

I verified those titles directly from the generated figures and from the notebook's stored
Plotly JSON. The notebook diff is large because those figure outputs were regenerated.
Token usage: total=395,554 input=356,798 (+ 8,301,952 cached) output=38,756 (reasoning 18,553)
To continue this session, run codex resume 019d234c-fa36-71a0-a475-baad97c49e
(eecs245) → notes git:(main) ×
(eecs245) → notes git:(main) × git status
On branch main
```

"Gradient field"

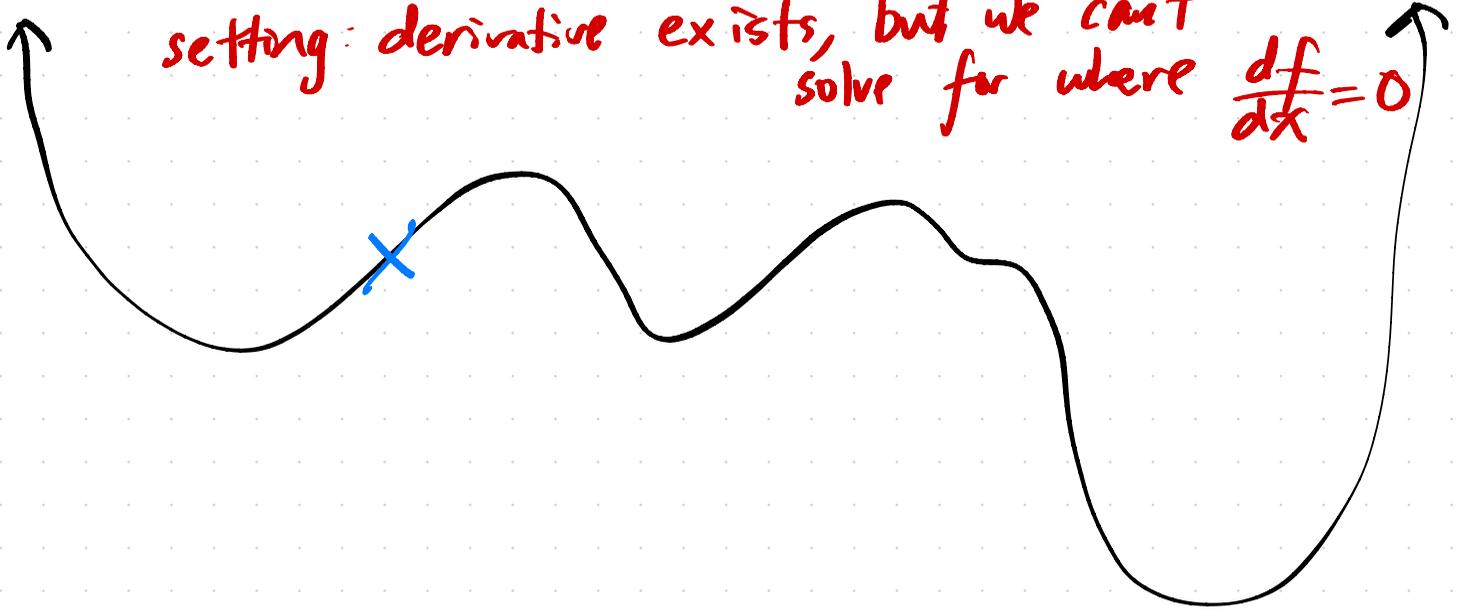
Gradient field of $f(\vec{x}) = x_1 e^{-x_1^2 - x_2^2}$



What's the point of gradient descent?

⇒ Minimize a function

setting: derivative exists, but we can't solve for where $\frac{df}{dx} = 0$



Gradient descent Goal: Find the \vec{x}^* that minimizes $f(\vec{x})$

Given an initial guess, $\vec{x}^{(0)}$
and a step size, $\alpha > 0$ (also called learning rate)

Update rule:

$$\vec{x}^{(t+1)} = \vec{x}^{(t)} - \alpha \nabla f(\vec{x}^{(t)})$$

opposite
direction from
gradient

gradient at
old guess

guess at iteration $t+1$

scalar

Terminate when $\|\nabla f(\vec{x}^{(t)})\| \leq 0.001$ (tolerance)

Activity 1, 8.3

$$f(\vec{x}) = (x_1 - 2)^2 + 2x_1 - (x_2 - 3)^2$$

$$\vec{x} \in \mathbb{R}^2$$

Given $\vec{x}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\alpha = \frac{1}{3}$

Find $\vec{x}^{(1)} = \vec{x}^{(0)} - \alpha \nabla f(\vec{x}^{(0)})$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} -2 \\ 6 \end{bmatrix} = \begin{bmatrix} 2/3 \\ -2 \end{bmatrix}$$

$$f(\vec{x}) = (x_1 - 2)^2 + 2x_1 - (x_2 - 3)^2$$

$$\nabla f(\vec{x}) = \begin{bmatrix} 2(x_1 - 2) + 2 \\ -2(x_2 - 3) \end{bmatrix}$$

$$\text{at } \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \nabla f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 2(0-2)+2 \\ -2(0-3) \end{bmatrix} = \begin{bmatrix} -2 \\ 6 \end{bmatrix}$$

Gradient descent is made for minimizing average loss to find optimal model parameters

$$R_{sq}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2$$

- don't need gradient descent to find \vec{w}^* :
there is a closed form sol'n,

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

- GD could be quicker if the data is massive

what if we used GD to minimize

$$R_{sq}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2 ?$$

$$\nabla R_{sq}(\vec{w}) = \frac{-2}{n} (X^T \vec{y} - X^T X \vec{w})$$

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \alpha \left(\frac{-2}{n} X^T \vec{y} - X^T X \vec{w}^{(t)} \right)$$

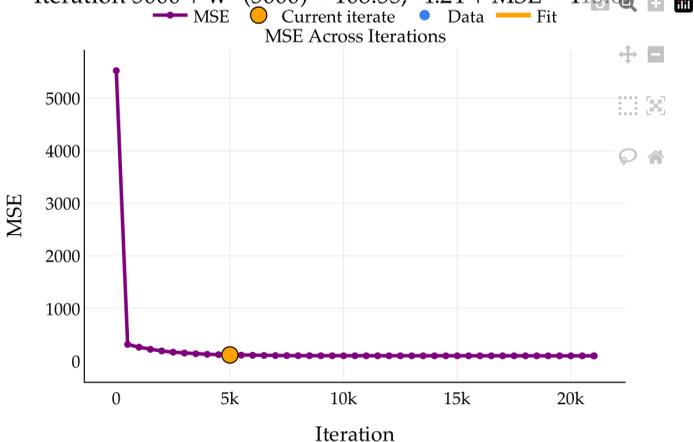
let's visualize: look at 8.4!

i = 0	$w^{(i)} = [0.000, 0.000]$	mse = 5523.5231	$ grad = 1229.842640$
i = 500	$w^{(i)} = [19.804, 6.102]$	mse = 314.8511	$ grad = 3.527944$
i = 1000	$w^{(i)} = [36.133, 4.200]$	mse = 260.7135	$ grad = 3.058219$
i = 1500	$w^{(i)} = [50.289, 2.551]$	mse = 220.0323	$ grad = 2.651035$
i = 2000	$w^{(i)} = [62.559, 1.121]$	mse = 189.4630	$ grad = 2.298065$
i = 2500	$w^{(i)} = [73.196, -0.118]$	mse = 166.4919	$ grad = 1.992091$
i = 3000	$w^{(i)} = [82.416, -1.193]$	mse = 149.2306	$ grad = 1.726856$
i = 3500	$w^{(i)} = [90.409, -2.124]$	mse = 136.2598	$ grad = 1.496935$
i = 4000	$w^{(i)} = [97.338, -2.931]$	mse = 126.5130	$ grad = 1.297627$
i = 4500	$w^{(i)} = [103.344, -3.631]$	mse = 119.1889	$ grad = 1.124856$
i = 5000	$w^{(i)} = [108.551, -4.237]$	mse = 113.6852	$ grad = 0.975088$
i = 5500	$w^{(i)} = [113.064, -4.763]$	mse = 109.5496	$ grad = 0.845261$
i = 6000	$w^{(i)} = [116.976, -5.219]$	mse = 106.4419	$ grad = 0.732719$
i = 6500	$w^{(i)} = [120.368, -5.614]$	mse = 104.1067	$ grad = 0.635162$
i = 7000	$w^{(i)} = [123.308, -5.957]$	mse = 102.3519	$ grad = 0.550594$
i = 7500	$w^{(i)} = [125.856, -6.254]$	mse = 101.0333	$ grad = 0.477285$
i = 8000	$w^{(i)} = [128.065, -6.511]$	mse = 100.0424	$ grad = 0.413738$
i = 8500	$w^{(i)} = [129.980, -6.734]$	mse = 99.2978	$ grad = 0.358651$
i = 9000	$w^{(i)} = [131.640, -6.928]$	mse = 98.7383	$ grad = 0.310899$
i = 9500	$w^{(i)} = [133.079, -7.095]$	mse = 98.3179	$ grad = 0.269504$
i = 10000	$w^{(i)} = [134.327, -7.241]$	mse = 98.0020	$ grad = 0.233621$
i = 10500	$w^{(i)} = [135.408, -7.367]$	mse = 97.7646	$ grad = 0.202516$
i = 11000	$w^{(i)} = [136.345, -7.476]$	mse = 97.5862	$ grad = 0.175552$
i = 11500	$w^{(i)} = [137.158, -7.571]$	mse = 97.4521	$ grad = 0.152179$
i = 12000	$w^{(i)} = [137.862, -7.653]$	mse = 97.3514	$ grad = 0.131917$
i = 12500	$w^{(i)} = [138.473, -7.724]$	mse = 97.2757	$ grad = 0.114353$

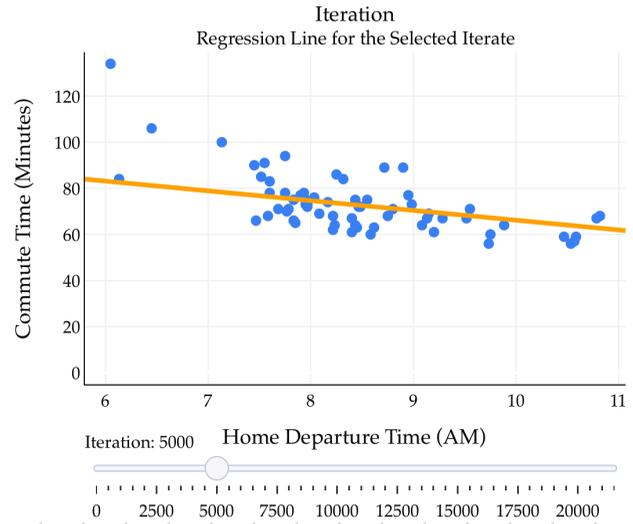
approaching 0

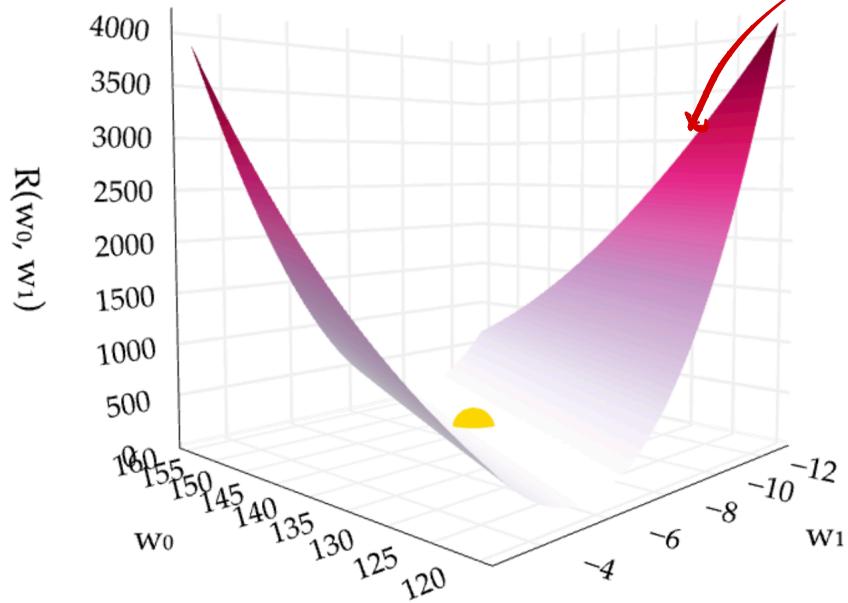


Iteration 5000 | $w^{(5000)} = 108.55, -4.24$ | MSE = 113.68

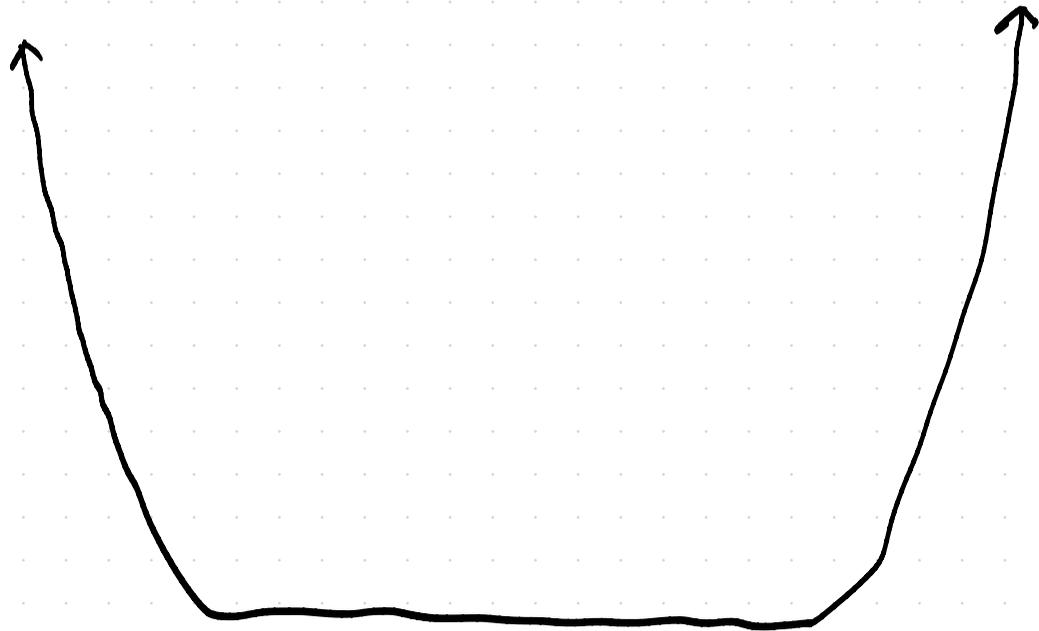


as t increases,
MSE approaches
theoretical minimum





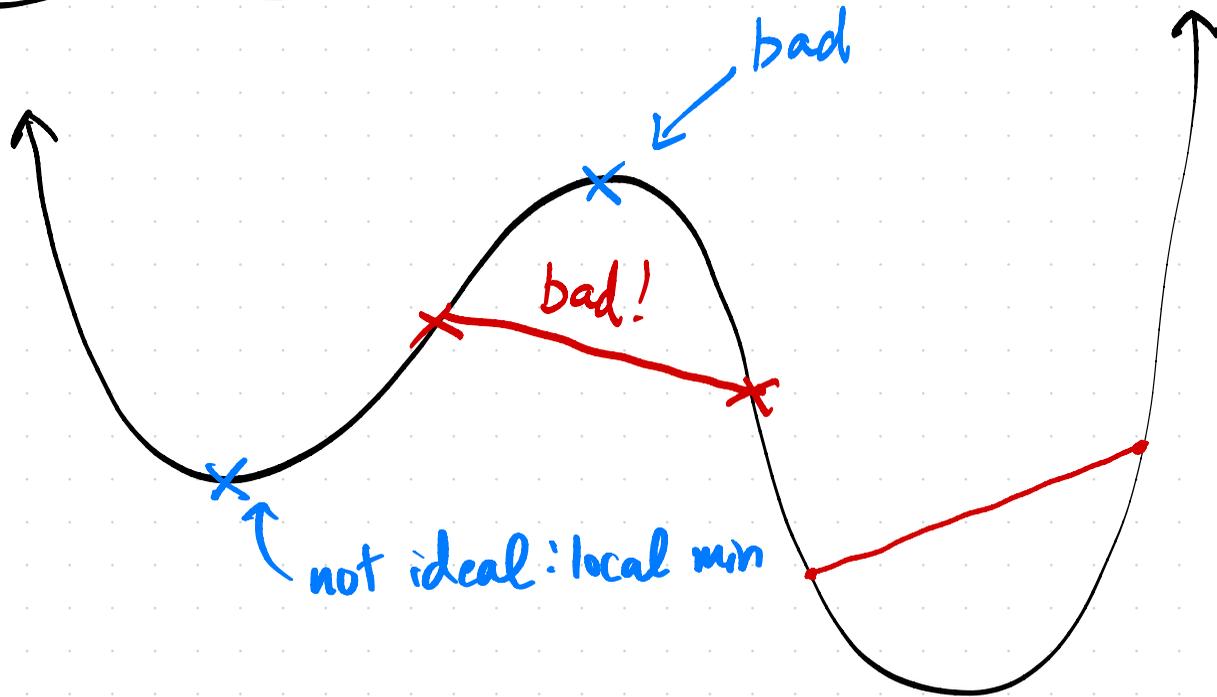
this is the
function we
just used
GD to
minimize!

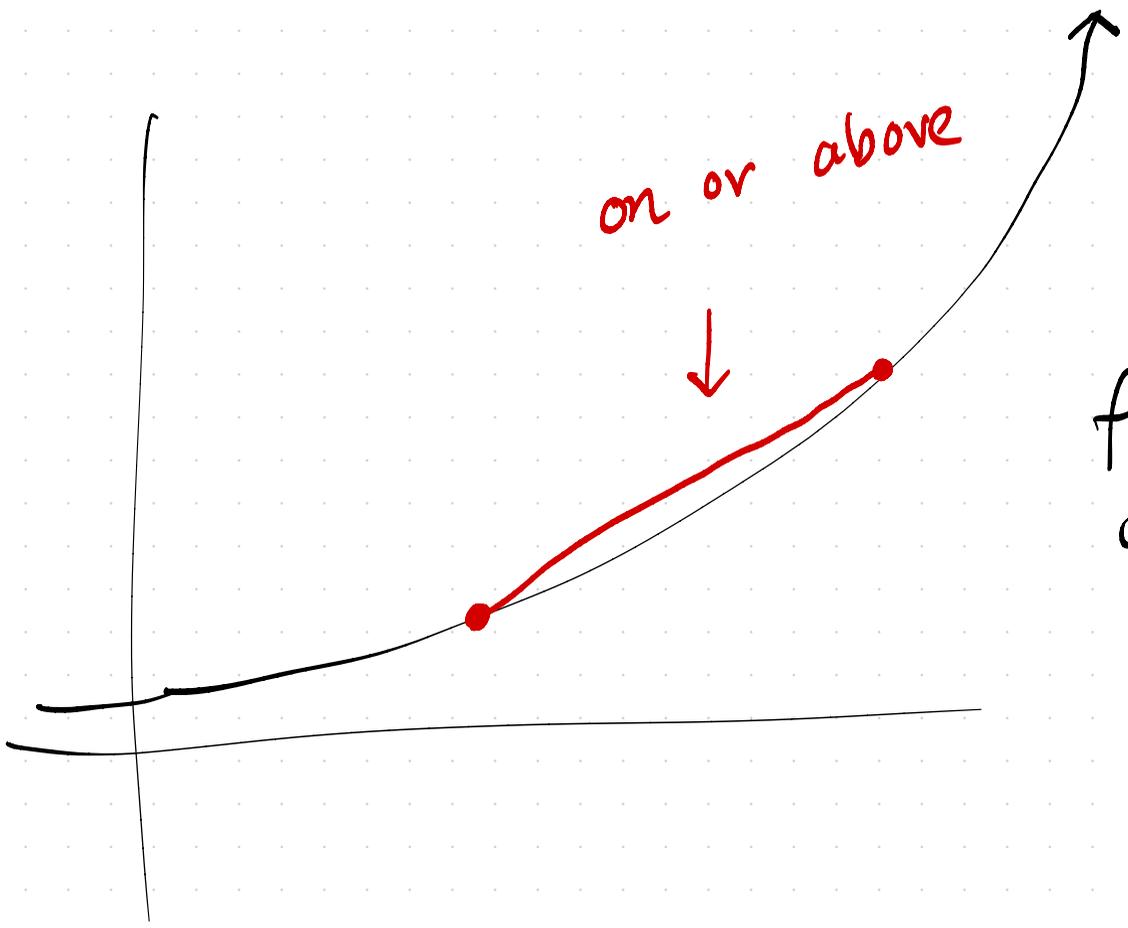


when would
 $R_{sq}(\vec{w})$ look
like this?

→ if X 's columns
are linearly
dependent

Is this function convex? no!



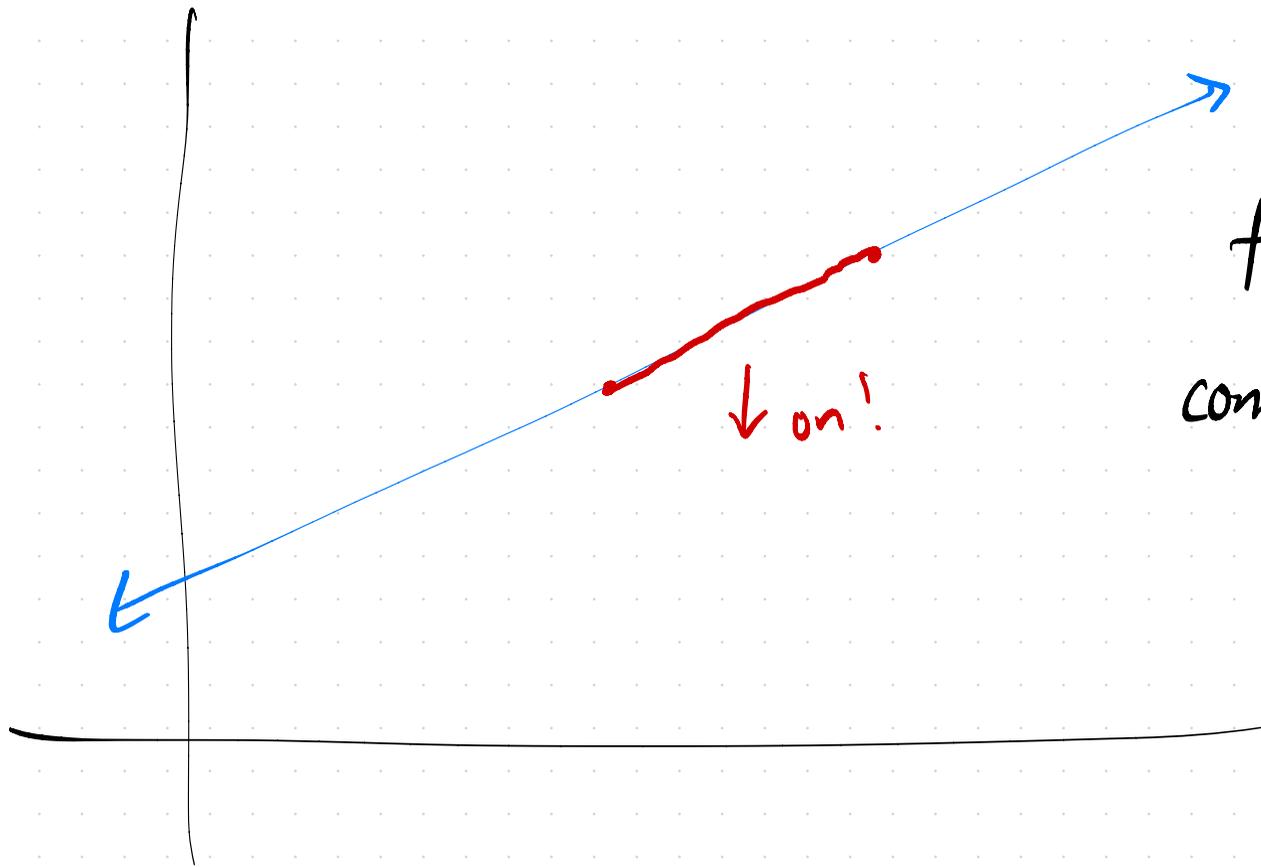


on or above



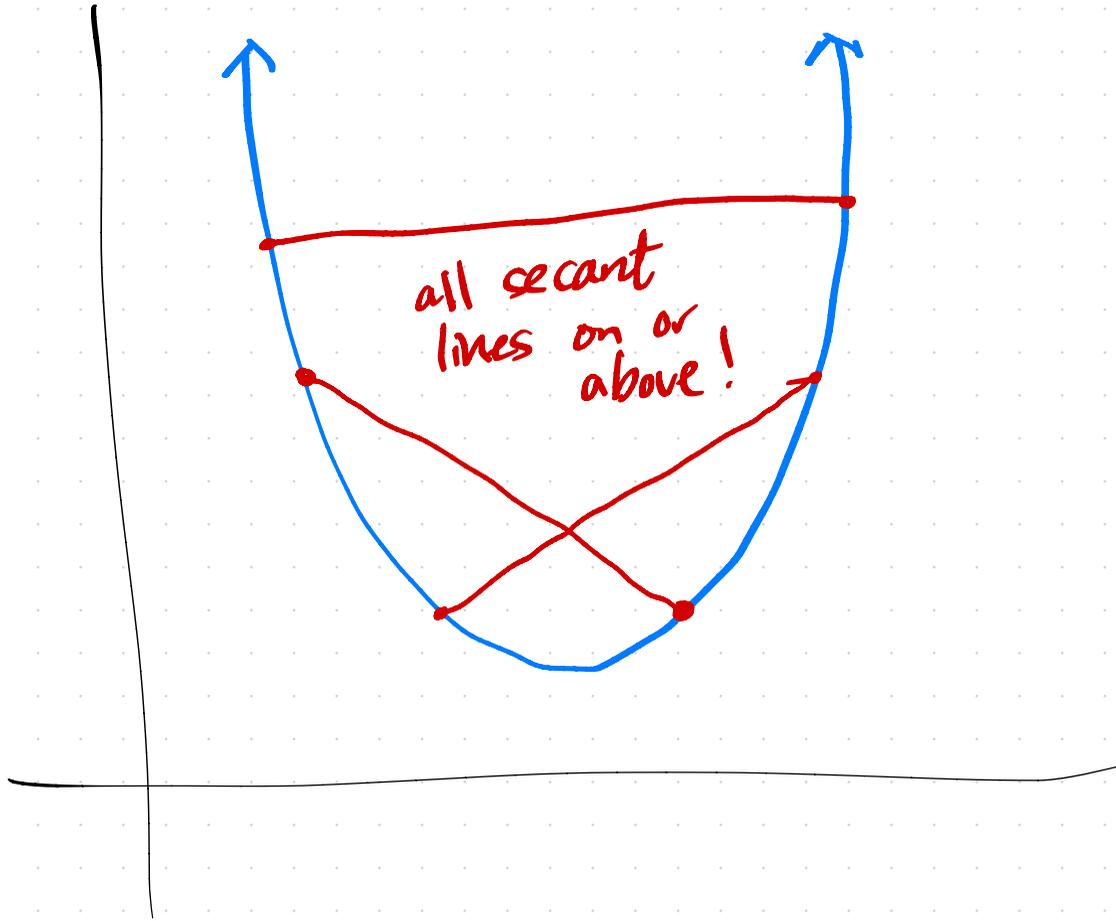
$$f(x) = e^x$$

convex? yes!



$$f(x) = 2x - 3$$

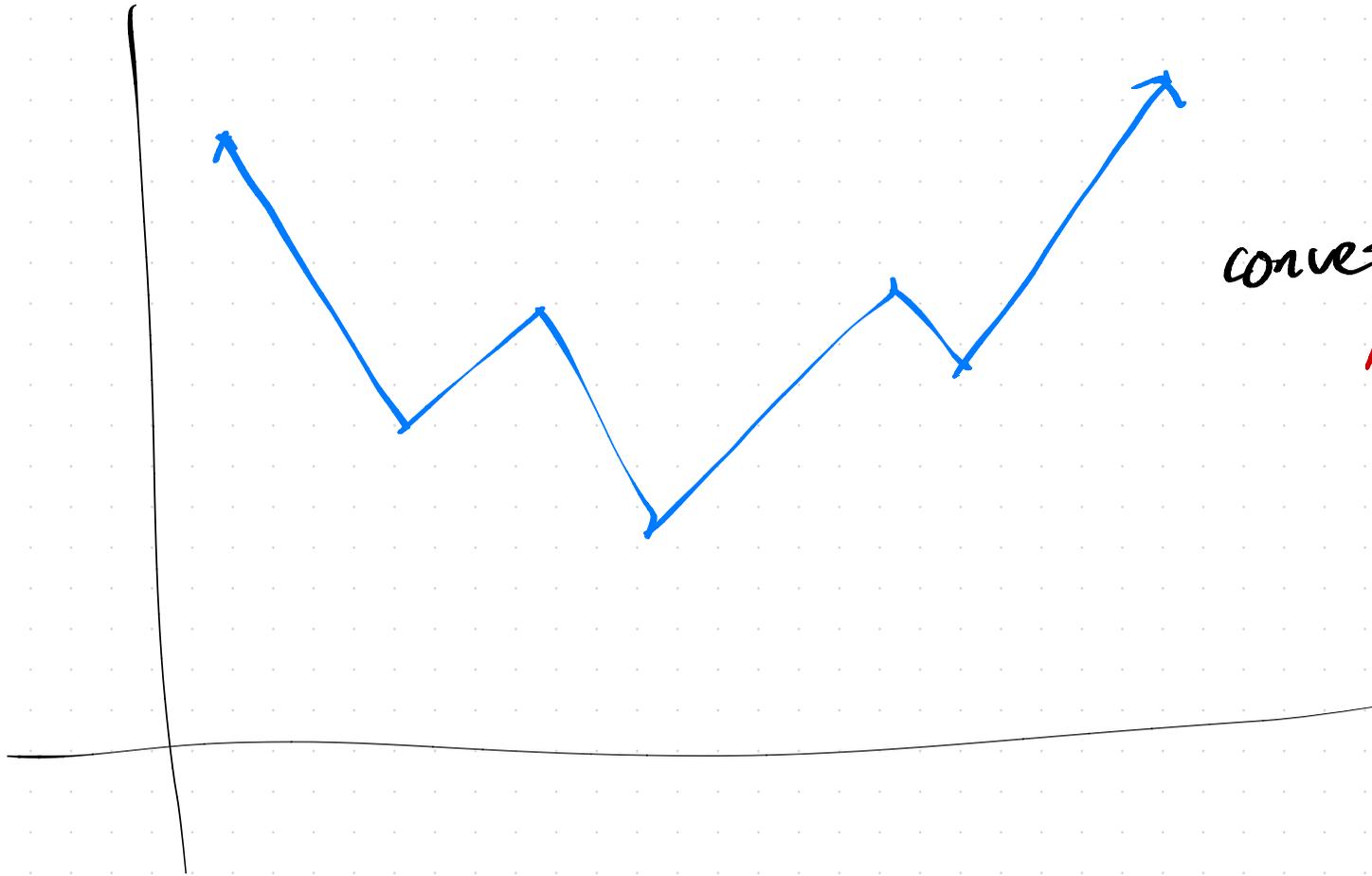
convex? yes!



$$f(x) = x^2 + 7$$

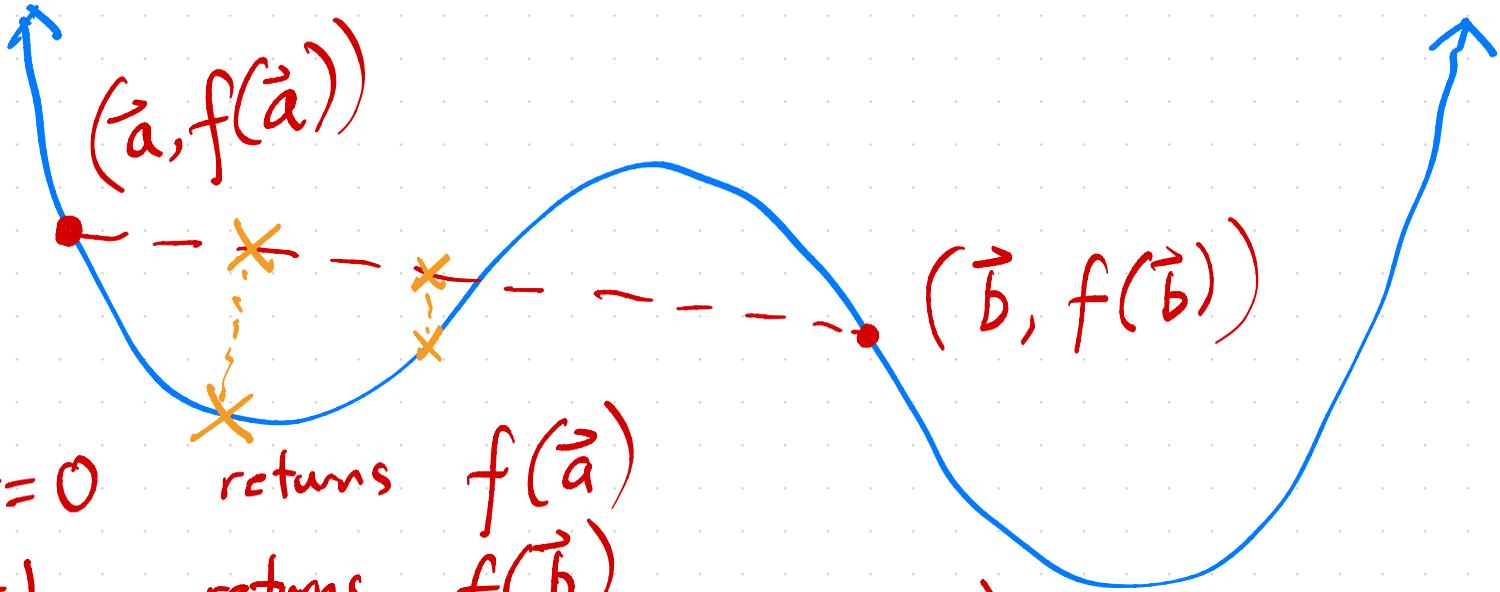
convex?

yes!



convex?
no!

Idea: f is convex if every possible secant line is on or above the curve



$t=0$ returns $f(\vec{a})$

$t=1$ returns $f(\vec{b})$

$$L(t) = f(\vec{a}) + t(f(\vec{b}) - f(\vec{a}))$$

$$= \underbrace{(1-t)} f(\vec{a}) + \underbrace{t} f(\vec{b})$$

"Formal definition"

$$f: \mathbb{R}^d \rightarrow \mathbb{R}$$

f is convex if for all \vec{a}, \vec{b} in its domain,
and any $t \in [0, 1]$,

$$f((1-t)\vec{a} + t\vec{b}) \leq (1-t)f(\vec{a}) + tf(\vec{b})$$

point on function
 $t\%$ of way from
 \vec{a} to \vec{b}

point on line $t\%$ of way
from $(\vec{a}, f(\vec{a}))$, $(\vec{b}, f(\vec{b}))$

This definition of convexity doesn't require derivatives; second derivative test for vector-to-scalar functions is more complicated

Why do we care? Convex functions play nice with gradient descent: no "traps" (local minima that aren't global minima)