Gradient Descent in action

Class starts @ 9:03 PM

- Biz-case deadline } 2 Weeks
- Biz-case discussion ✓

Agenda:

Math: $f(x)$

1. Gradient Descent code + experiments
2. Linear Regression as optimization → ML-1 4-d
3. Revisit the classification problem

```
import random
import math
import numpy
```

0s

$$\frac{dx^n}{dx} = nx^{n-1}$$

Math:

one-variable

$$f(x) = x^2 + 2x^1 + 10$$

```
#define f(x)
def f(x):
  return math.pow(x,2)+2*x+10

# df_dx; we could have used the limits concepts form prev class too.
def derF(x):
  return 2*x+2
```

$$\min_x f(x)$$

```
def gradientDescentUnivariate():

  x0 = random.random()
  i=0 # iterations
  eta = 0.1 # learning rate

  xi = x0
  df_dx = derF(xi) # df_dx at x0

  while abs(df
```

$$\frac{df(x)}{dx} = 2x + 2 \cdot 1 + 0$$

$$= 2x + 2$$

```
    return 2*x+2
```

```
def gradientDescentUnivariate():

    x0 = random.random()
    i=0 # iterations
    eta = 0.1 # learning rate

    xi = x0
    df_dx = derF(xi) # df_dx at x0

    while abs(df_dx - 0.0) > 0.001: # threshold = 0.001
        print(i)
        print(xi)
        print(df_dx)
        print("************************")

        df_dx = derF(xi) # compute the derivative
        xi = xi - eta*df_dx # update equation
        i = i+1

    return xi
```

10^{-3}

$\min_x f(x)$

x_0 : randomly

x_1

x_2

x_3

$\vdots$

df_dx

$x_{i+1} = x_i - \eta \frac{df}{dx}$ x_i

```
    return 2*x+2
```

```
def gradientDescentUnivariate():

    x0 = random.random()
    i=0 # iterations
    eta = 0.1 # learning rate

    xi = x0
    df_dx = derF(xi) # df_dx at x0

    while abs(df_dx - 0.0) > 0.001: # threshold = 0.001
      print(i)
      print(xi)
      print(df_dx)
      print("************************")

      df_dx = derF(xi) # compute the derivative
      xi = xi - eta*df_dx # update equation
      i = i+1

    return xi
```

```
i=0 # iterations
eta = 0.1 # learning rate

xi = x0
df_dx = derF(xi) # df_dx at x0

while abs(df_dx - 0.0) > 0.001: # threshold = 0.001
  print(i)
  print(xi)
  print(df_dx)
  print("************************")

  df_dx = derF(xi) # compute the derivative
  xi = xi - eta*df_dx # update equation
  i = i+1

return xi
```

$$f_{min} = \min_x f(x)$$

$$x_{min} = x^* = \tilde{\text{argmin}}_x f(x)$$

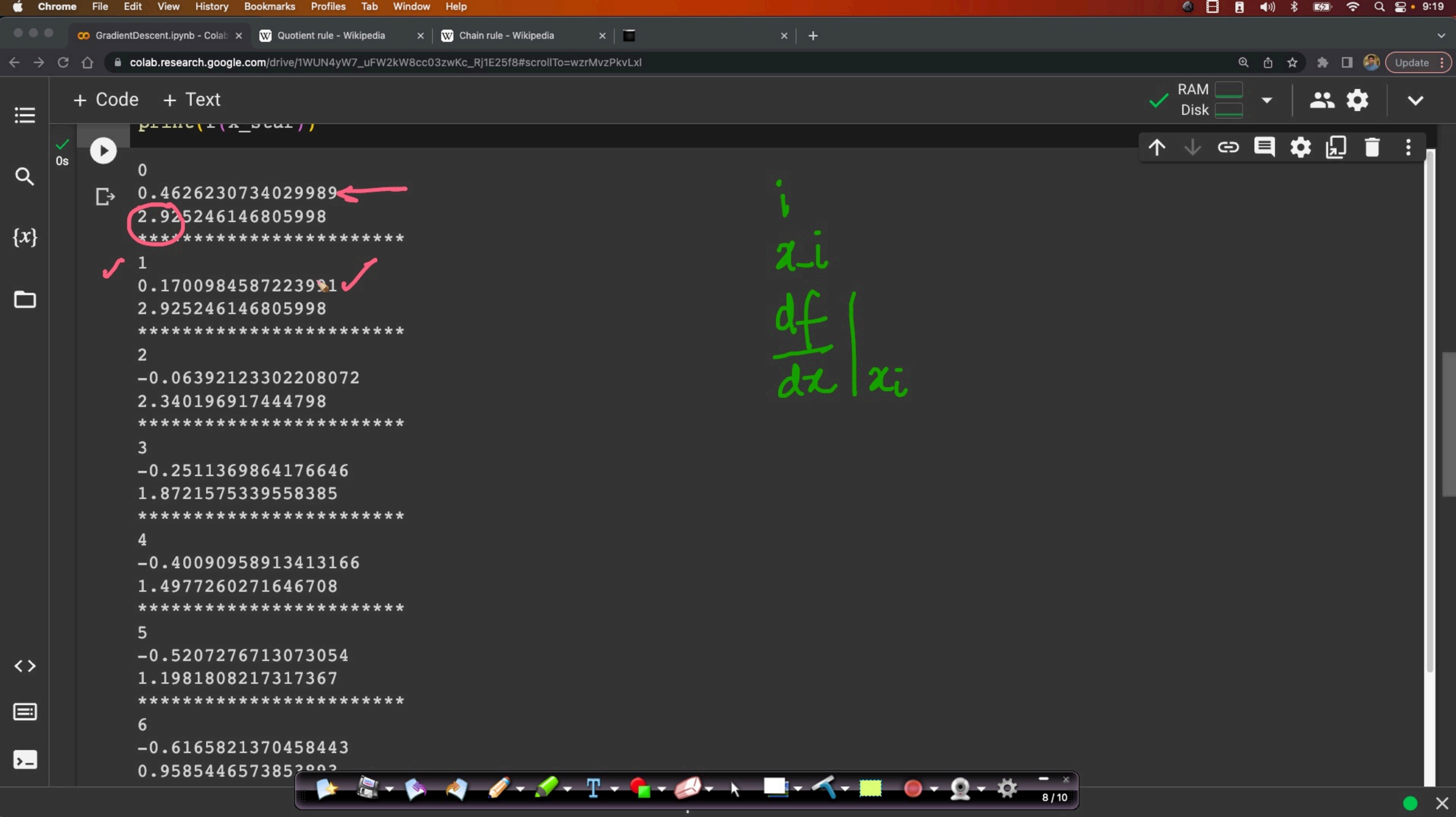
```
x_star = gradientDescentUnivariate()
print(x_star)
print(f(x_star))
```

```
0
0.7326180598960425
3.4652361197920847
```

```
i=0 # iterations
eta = 0.1 # learning rate

xi = x0
df_dx = derF(xi) # df_dx at x0

while abs(df_dx - 0.0) > 0.001: # threshold = 0.001
  print(i)
  print(xi)
  print(df_dx)
  print("************************")

  df_dx = derF(xi) # compute the derivative
  xi = xi - eta*df_dx # update equation
  i = i+1

return xi
```

```
x_star = gradientDescentUnivariate()
print(x_star)
print(f(x_star))
```

```
0
0.7326180598960425
3.4652361197920847
```


Chrome File Edit View History Bookmarks Profiles Tab Window Help
GradientDescent.ipynb - Colab
Quotient rule - Wikipedia
Chain rule - Wikipedia
colab.research.google.com/drive/1WUN4yW7_uFW2kW8cc03zwKc_Rj1E25f8#scrollTo=wzrMvzPkvLxl
Update
+ Code + Text
RAM
Disk
0s
0
0.4626230734029989
2.925246146805998
****
1
0.1700984587223991
2.925246146805998
****
2
-0.06392123302208072
2.340196917444798
****
3
-0.2511369864176646
1.8721575339558385
****
4
-0.40090958913413166
1.4977260271646708
****
5
-0.5207276713073054
1.1981808217317367
****
6
-0.6165821370458443
0.9585446573852893
i
x_i
df/dx | x_i
8 / 10

Chrome File Edit View History Bookmarks Profiles Tab Window Help

GradientDescent.ipynb - Colab | Quotient rule - Wikipedia | Chain rule - Wikipedia

colab.research.google.com/drive/1WUN4yW7_uFW2kW8cc03zwKc_Rj1E25f8#scrollTo=wzrMvzPkvLxl

+ Code + Text

RAM Disk

```
[3]    return xi
```

```
x_star = gradientDescentUnivariate()
print(x_star)
print(f(x_star))
```

```
0.05269654982688987
**********
20
-0.9831371040553952
0.042157239861511986
**********
21
-0.9865096832443162
0.03372579188920954
**********
22
-0.989207746595453
0.02698063351136759
**********
23
-0.9913661972763623
0.021584506809094073
**********
24
-0.9930929578210899
```

$$x_{22} = x_{21} - \eta \left.\frac{\partial f}{\partial x}\right|_{x_{21}}, \quad \eta = 0.1$$

9 / 11

0s

```
0.002897023463917181
**********
33
-0.9990729524915465
0.0023176187711337892
**********
34
-0.9992583619932371
0.0018540950169070314
**********
35
-0.9994066895945897
0.001483276013525714
**********
36
-0.9995253516756717
0.00118662081082066
**********
-0.9996202813405374
9.000000144186261
```

$-1 = x^+$

$f(x)$

$\min_x f(x) = 9$

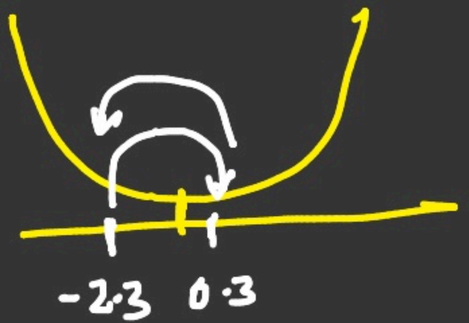
$\arg\min_x f(x) = -1$

```
19994
0.30312655233365504
-2.60625310466731
**********
19995
-2.303126552333655
2.60625310466731
**********
19996
0.30312655233365504
-2.60625310466731
**********
19997
-2.303126552333655
2.60625310466731
**********
19998
0.30312655233365504
-2.60625310466731
**********
19999
-2.303126552333655
2.60625310466731
**********
20000
0.30312655233365504
-2.60625310466731
**********
```

$-x_i$

$\rightarrow x_{i+1}$

$\rightarrow x_{i+2}$



0.001

$\eta = 0.1 \rightarrow 36$

$\eta = 0.3 \rightarrow 9$

$\eta = 0.5 \rightarrow 1$

$\eta = 1.0 \rightarrow \infty$ (oscillation problem)

[hack: $\eta_{i+1} = \eta_i + 0.9$

Q

9:39PM

$$f(x,y) = x^2 + y^2 + 2xy + 5x + 3y + 2$$

obj: find x^* & y^* that

$$\min_{x,y} f(x,y)$$

(1)

$$\frac{\partial f}{\partial x} = 2x + 2y + 5 \qquad \frac{\partial f}{\partial y} = 2y + 2x + 3$$

pseudo code:

x_i = random - number

y_i = random - number

$i = 0$; $\eta = 0.1$

✓ df_dx = derFx (xi)

✓ df_dy = derFy (yi)

while ((abs (df_dx) > 0.001) OR (|df_dy| > 0.001))

{

df_dx = derFx (xi)

df_dy = derFy (yi)

}

$x_i : \frac{\partial f}{\partial x} \approx 0$

$y_i : \frac{\partial f}{\partial y} \approx 0$

{ NOT BOUNDARY cases

$$
\begin{cases}
x_i = x_i - \eta\left(\frac{\partial f}{\partial x}\right) \\
y_i = y_i - \eta\left(\frac{\partial f}{\partial y}\right) \\
i = i + 1
\end{cases}
$$

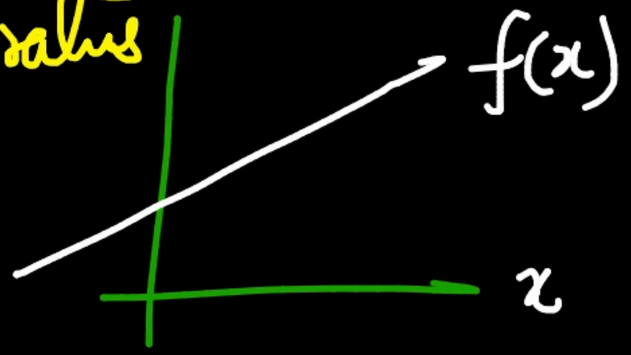
$\rightarrow \cdots$

return (x_i, y_i)

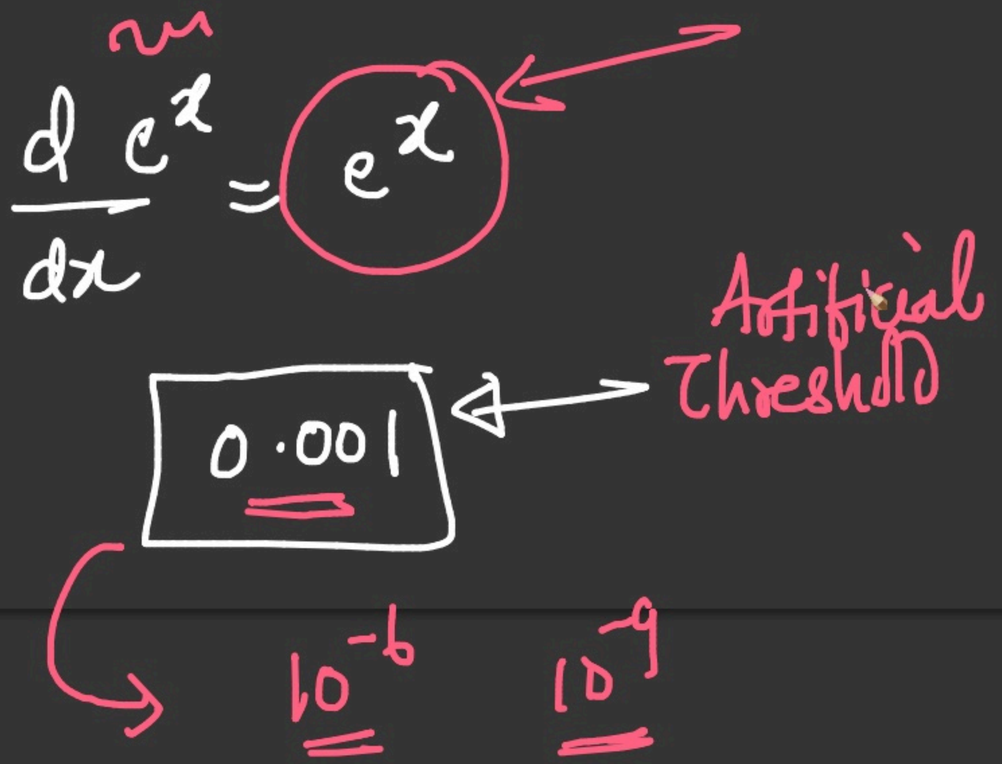
Boundary Case

(last-class) → perturbation test

1. saddle point → (x_i, y_i)
2. no minima → 100k iterations
3. oscillation → x_i, x_{i+1}, x_{i+2}
4. 0.001: threshold → $1e^{-6}$ (or) $1e^{-9}$



```
80982
-6.69756756696005
0.0012339248835084062
**********
80983
-6.697579906056629
0.001233909657896161
**********
80984
-6.697592245000956
0.0012338944326596554
**********
80985
-6.697604583793034
0.0012338792077988753
**********
80986
-6.697616922432867
0.001233863983313807
```

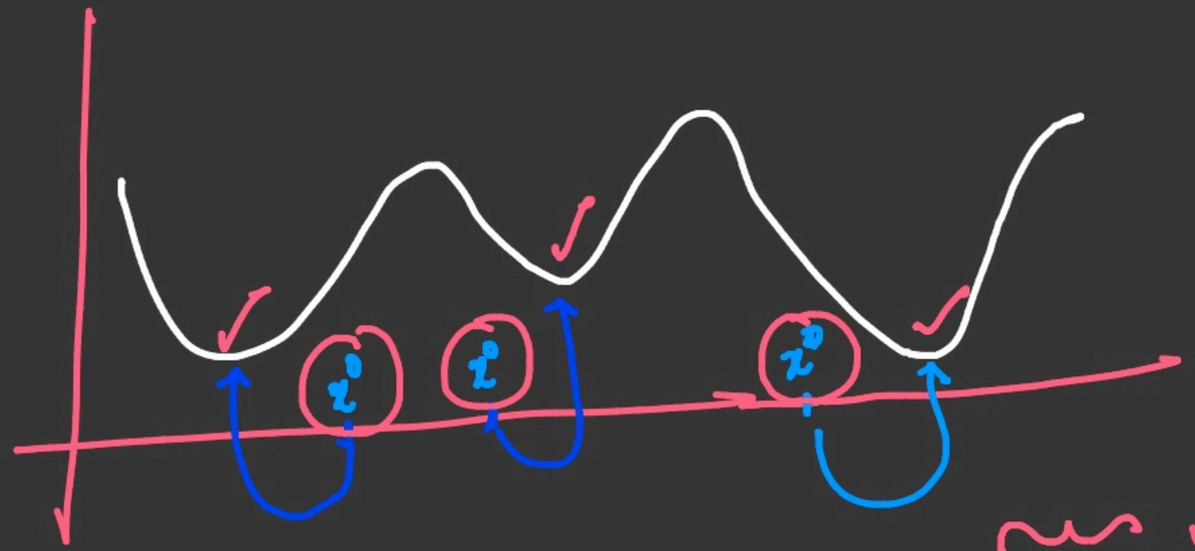


$$\frac{\partial f}{\partial x} \to 0 \qquad \frac{\partial f}{\partial y} \neq 0$$

$$x_{i+1} = x_i - \eta \frac{\partial f}{\partial x}$$

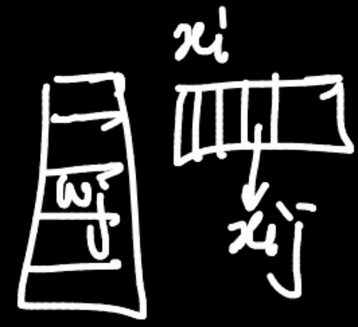
$$\therefore y_{i+1} = y_i - \eta \frac{\partial f}{\partial y}$$

```
**********
99936
-6.907728433272202
0.0010000368467572279
**********
99937
-6.907738433540662
0.0010000268460702846
**********
99938
-6.907748433709118
0.0010000168455833601
**********
99939
-6.9077584337775715
0.0010000068452964478
**********
-6.907768433746024
0.0009999868453226368
```



Classfn

$$\text{Max} \sum_{i=1}^{n} \frac{w^T x_i + b}{\|w\|} \cdot y_i$$



$$\min_{w_j, b} \; -\sum_{i=1}^{n} \left(\frac{\sum_{j=1}^{d} w_j x_{ij} + b}{\|w\|} \right) \cdot y_i \longrightarrow \mathcal{L}(w_1, w_2 \cdots w_d, b)$$

d+1 Variables

derivatives :

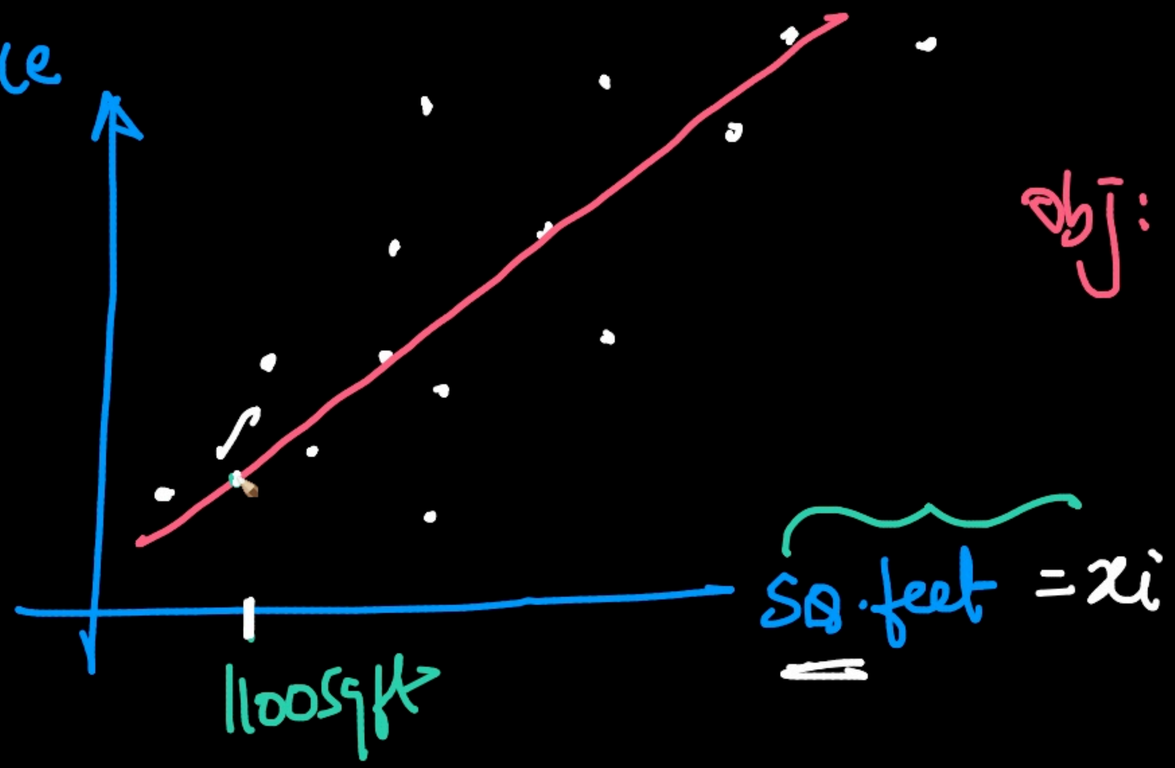
$$\frac{\partial L}{\partial w_j} \quad \text{for all } j = 1 \text{ to } d$$

$$\frac{\partial L}{\partial b}$$

$$\frac{d}{dw_j}\left[\sum_{j=1}^{d} w_j \cdot x_{ij} + b\right] \Big/ \sqrt{\sum_{j=1}^{d} w_j^2}$$

→ Cumbersome but doable

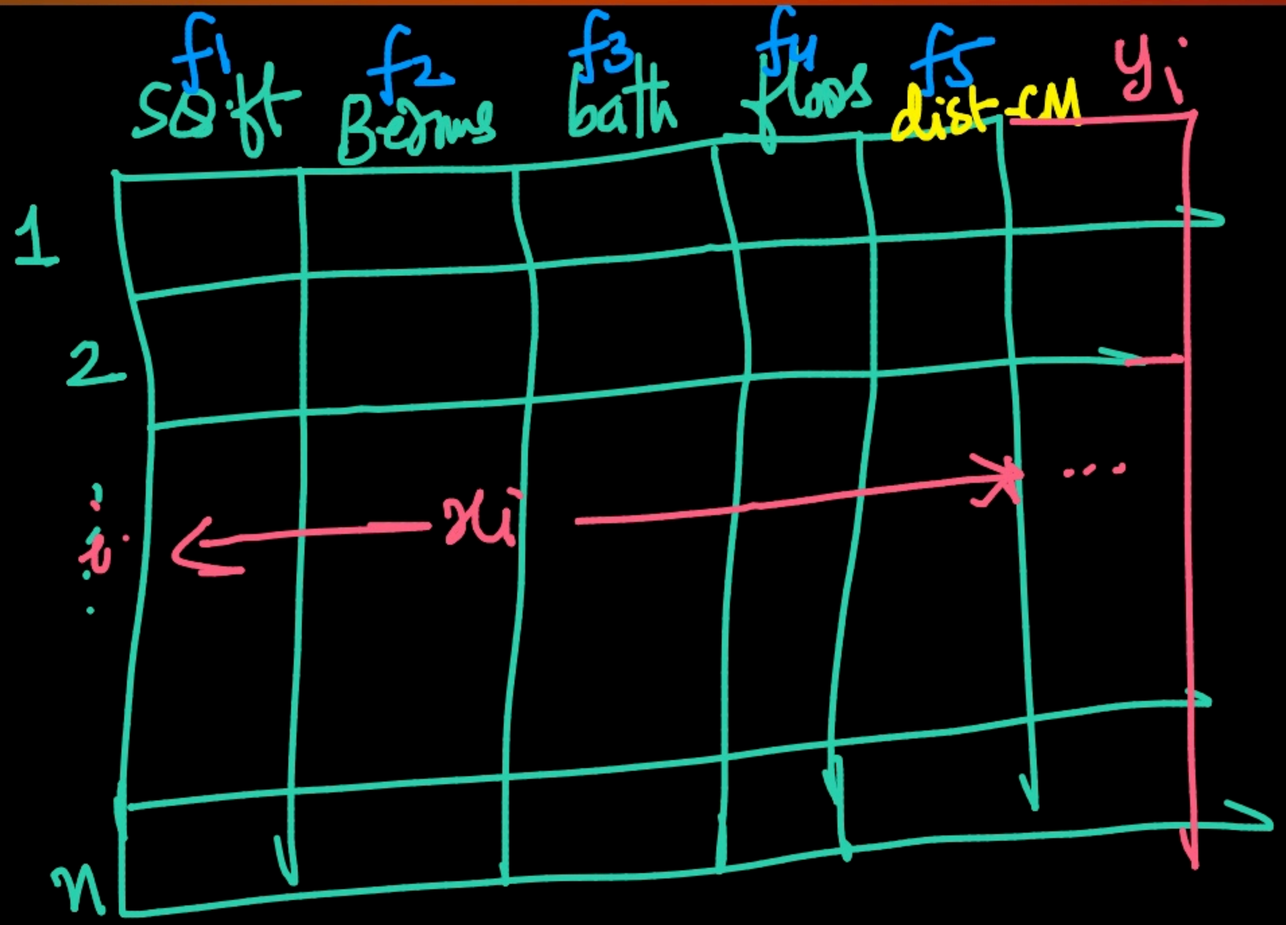
Linear Regression

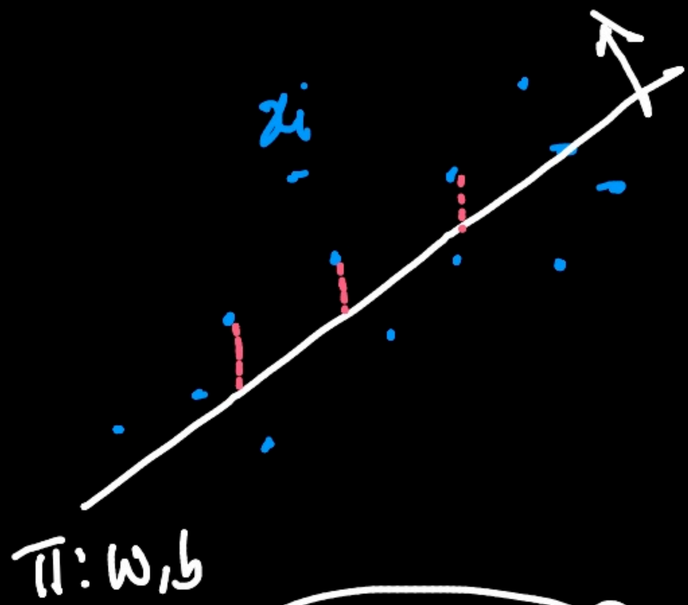


obj: line that is as close as possible to given data

$i = 1$ to n

$\{ x_i, y_i$





Let $\|w\| = 1$

$$y_i - w^T x_i + b$$

$$\min_{w,b} \sum_{i=1}^{n} \left(y_i - (w^T x_i + b)\right)^2$$

y_i: actual, $(w^T x_i + b)$: pred

$$\mathcal{L}(w,b)$$

$$\sum_{i=1}^{n} y_i - (w_1 x_{i1} + w_2 x_{i2} + \dots + w_j x_{ij} + \dots + w_d x_{id} + b)^2$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{n} (-x_{ij}) \; 2\left(y_i - (w^T x_i + b)\right)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} (-1) \cdot 2\left(y_i - (w^T x_i + b)\right)$$

1 2 3 ---- d

x_i : : : → y_i price of a home

$y_i = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} \cdots + w_d x_{id} + b$

w, b $\quad$ x_i

pred-value: $w^T x_i + b$

Recap:

1. $f(x)$: one variable → code
2. $f(x, y)$ → 2-var → pseudo-code + boundary cases
3. $\mathcal{L}(w_1, w_2 \ldots w_n, b)$ → ✓
4. Linear Regr → optmzn

ML

$\sum_{i=1}^{n}$: Computationally expensive

Next class:

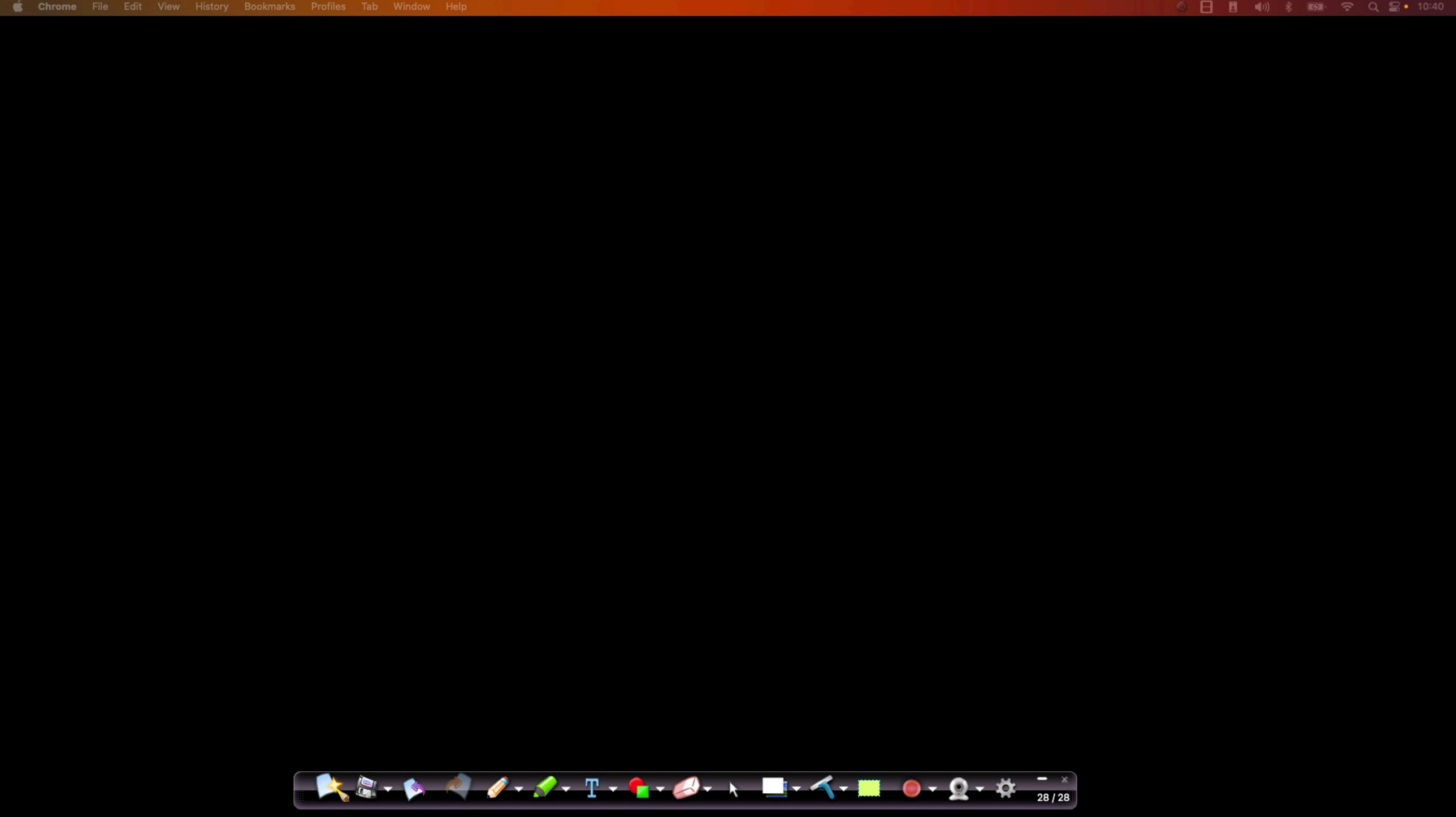
$$\min_{w,b} \mathcal{L}(\vec{w}, b) \quad \text{s.t} \quad \|w\| = 1$$

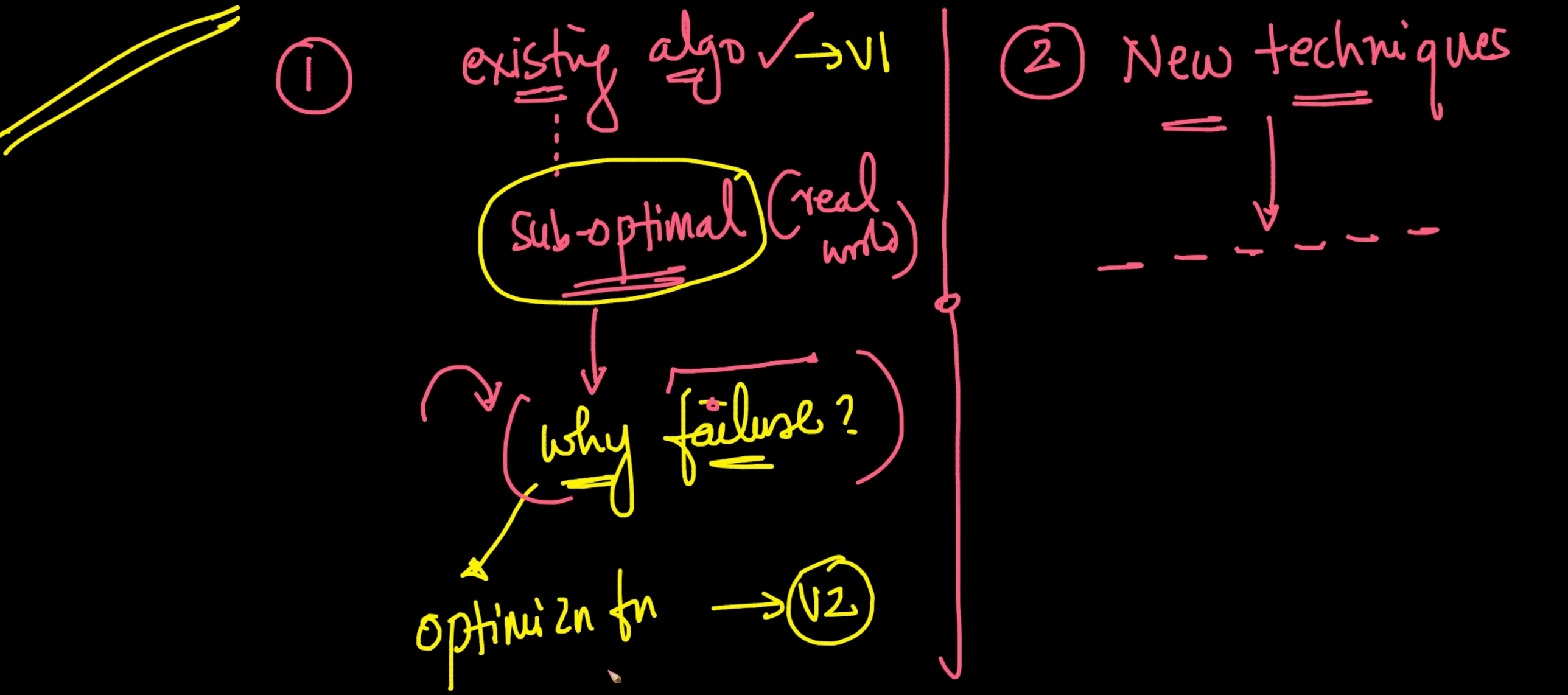
($\vec{w}$: vec; $\mathcal{L}(\vec{w}, b)$: scalar)

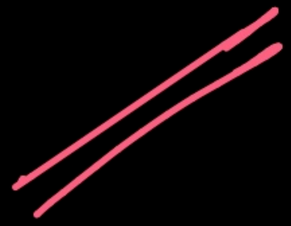
Classfn

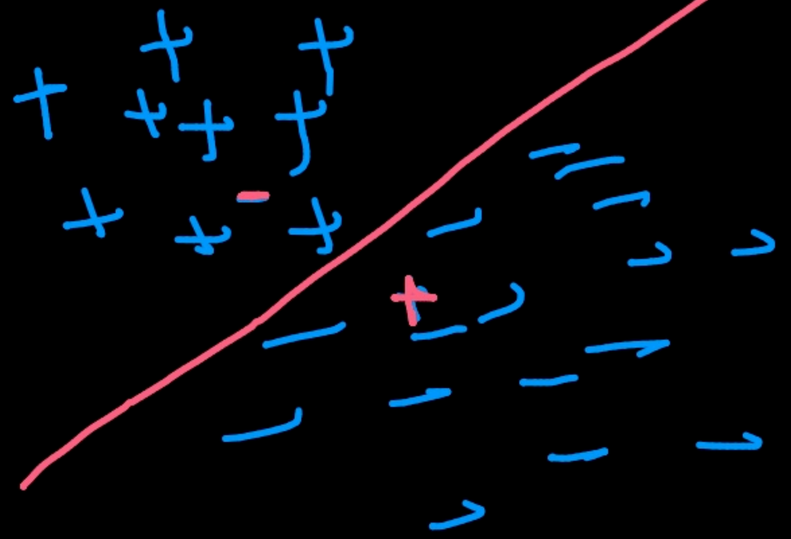
Constrained optmzn

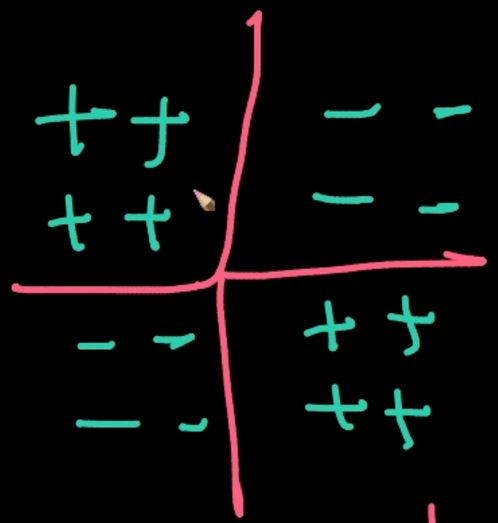
PCA


Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:40
28 / 28


1
existing algo ✓ → V1
Sub-optimal (real world)
why failure ?
optimization fn → V2
2
New techniques







non-linearly sep

2m

```
**********
99936
-6.907728433272202
0.0010000368467572279
**********
99937
-6.907738433540662
0.0010000268460702846
**********
99938
-6.907748433709118
0.0010000168455833601
**********
99939
-6.9077584337775715
0.0010000068452964478
**********
-6.907768433746024
0.0009999868453226368
```