Agentic AI?

→ planner + executor + memory + verify.

LLM → large language models

Basic ML

→ Regression, Classification

↓ (continuous)

↳ classify

↳ Supervised & Unsupervised

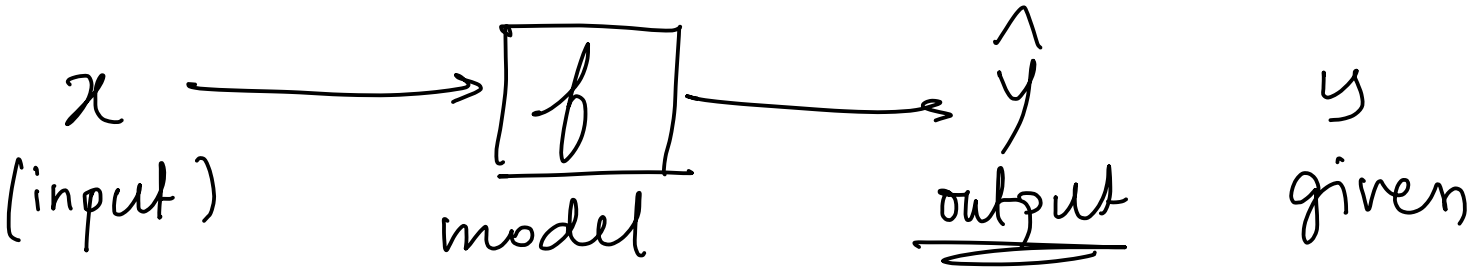
↓ ↓

input ⟷ output X output

Q] Price of a house :-

features :- size, location, floors

output :- price



Linear Regression

$$f_{w,b}(x) = wx + b = [\hat{y} \longleftrightarrow y]$$

$x \rightarrow$ size house $\longleftrightarrow$ price $\rightarrow$ (m pairs)

Error fxn $\rightarrow \sum (\hat{y}_i - y)^2$

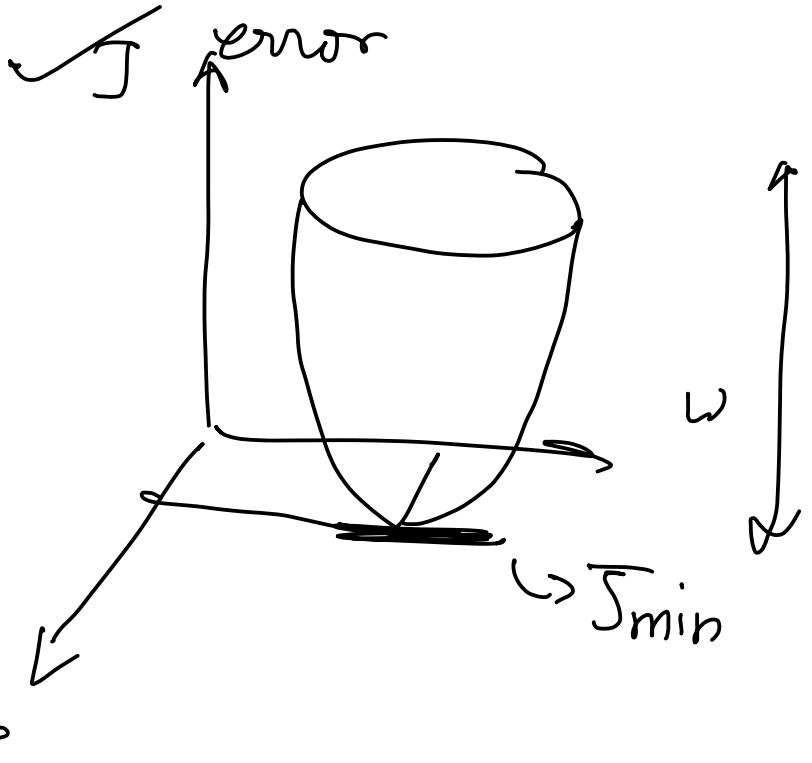
$\rightarrow \frac{\sum (\hat{y}_i - y)^2}{2m}$? 2m?

↳ $J(w, b)$

$$J(w,b) = \frac{1}{2m} \sum \left((w_i x_i + b_i) - y \right)^2$$

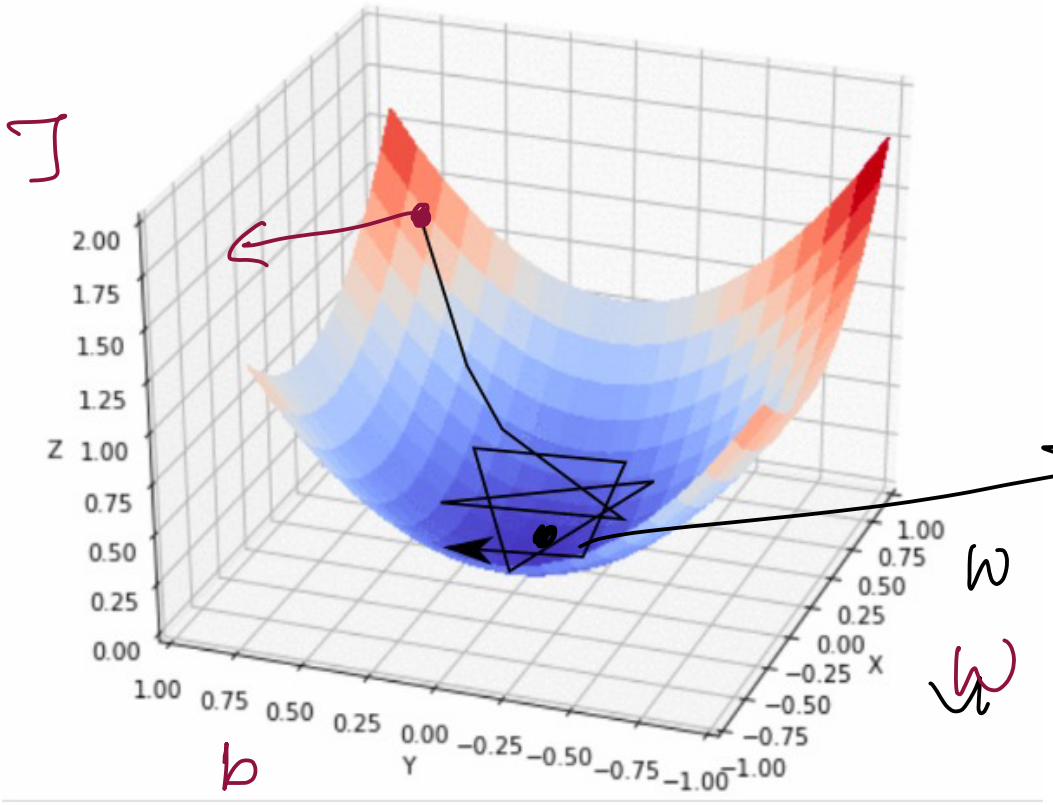
$J \rightarrow w, b$ quadratic

squared error



$y \approx \hat{y}$

Gradient descent



'adjusting value.' → updating

$$w' = w - \alpha \cdot \frac{\partial}{\partial w} J(w,b)$$

α → learning rate

$$b' = b - \alpha \cdot \frac{\partial J(w,b)}{\partial b}$$

x → 1 feature size → 1 output y price

Multiple features

$$f_{w,b}(x) = w_1 x_1 + w_2 x_2 + \dots + b$$

$$= (\vec{w} \cdot \vec{x} + b) \longrightarrow \text{vectorization}$$

(1)

$J(\vec{w}, b)$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(w_1, \dots w_n, b)$$

w_1
$\vdots$
w_n
b

⟶ Price? size, location, floor

100 – 1000 sq

place → coded num

1, 2

Feature Scaling

100 ⟶ 1000 , 1, 2, ... 10

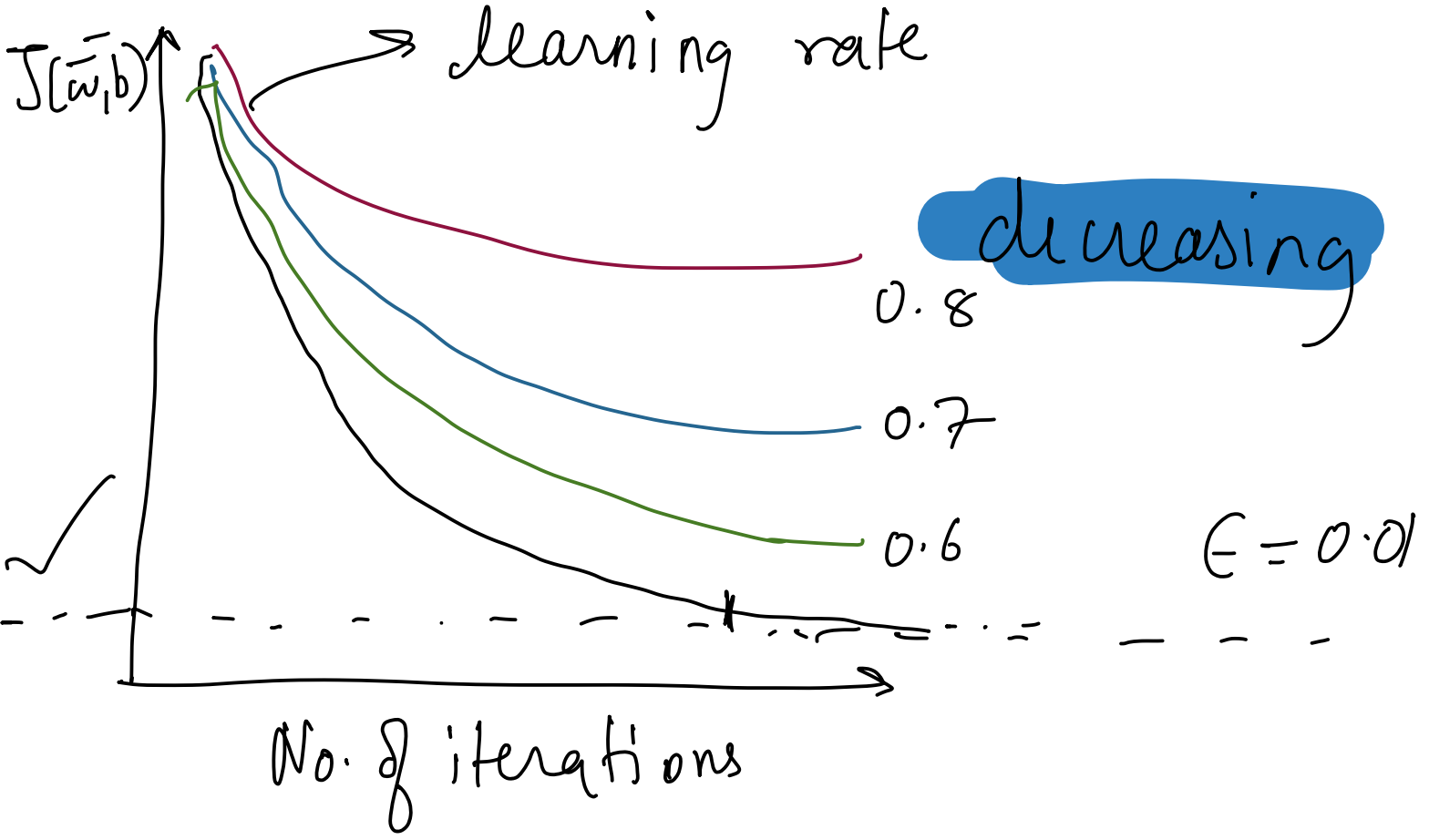
$0 \rightarrow 1$, $-1 \rightarrow 1$

1) Normalization

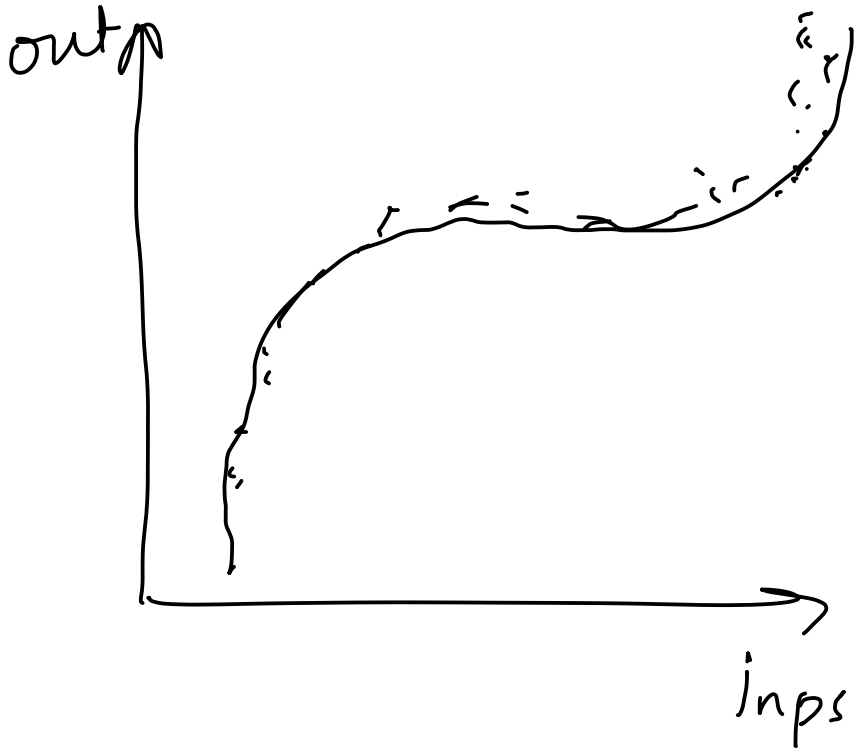
$x_1 \quad x_1^1 \; x_1^2 \; \ldots \; x_1^n$

$$\bar{x}_1^i = \frac{x_1^i - \mu_1}{\sigma_1} \qquad x_2^i = \frac{x_2^i - \mu_2}{\sigma_2}$$

Learning curve



Polynomial regression



2D

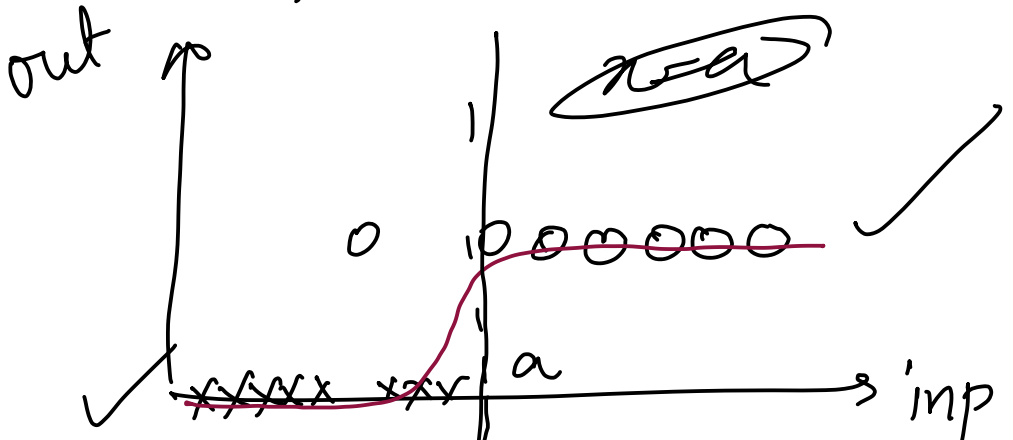
$$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + b$$

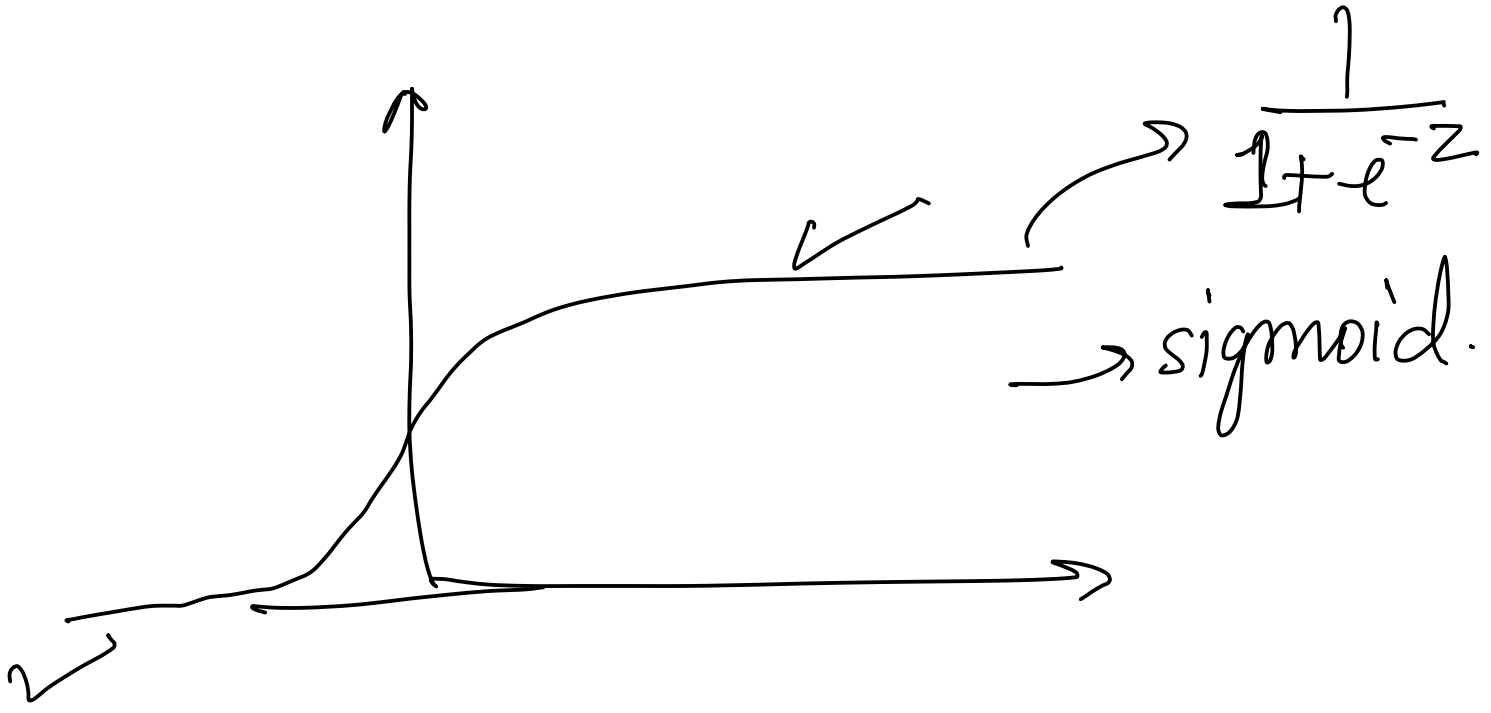
1 feature

logistic regression

→ classification task, T/F

→ output → 1, 0





$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b = z$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

decision boundary

$$f_{\vec{w},b}(\vec{x}) \geq a \longrightarrow y = 1$$

$$< a \longrightarrow y = 0$$

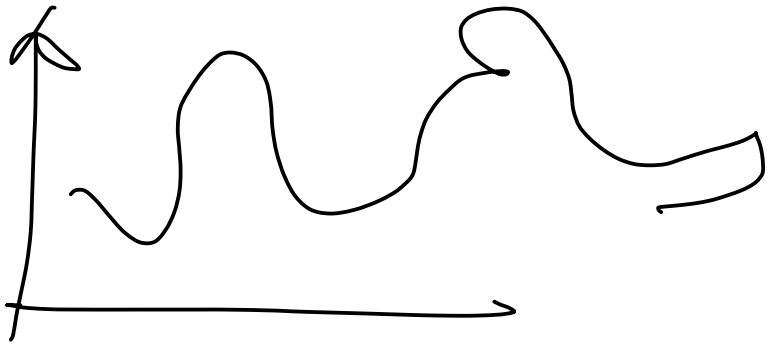
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(w_1 a_1 + w_2 a_2 + b)$$

$$(z = x_1 + 2x_2 - 3 = 0) \rightarrow \text{decision boundary}$$

Cost fun

$$J = \frac{1}{1 - e^{-(\vec{w}\cdot\vec{x}+b)}} \; y \rightarrow$$

~~quad J~~



New cost fun

→ logistic loss fun

$$L(f_{w,b}(x,y)) = \begin{cases} -\log(f_{w,b}(x_i)) & y=1 \\ -\log(1 - f_{\vec{w}b}(x_i)) & y=0 \end{cases}$$

$$= -y^{(i)} \log\left(f_{w,b}(x^{(i)})\right) - (1-y^{(i)}) \log\left(1-f_{\vec{w},b}(x^{i})\right)$$

$$J(\vec{w},b) = -\frac{1}{m} \sum L\left(f_{w,b}(x^{i}, y^{i})\right)$$

- $w_j^{\circ} = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$

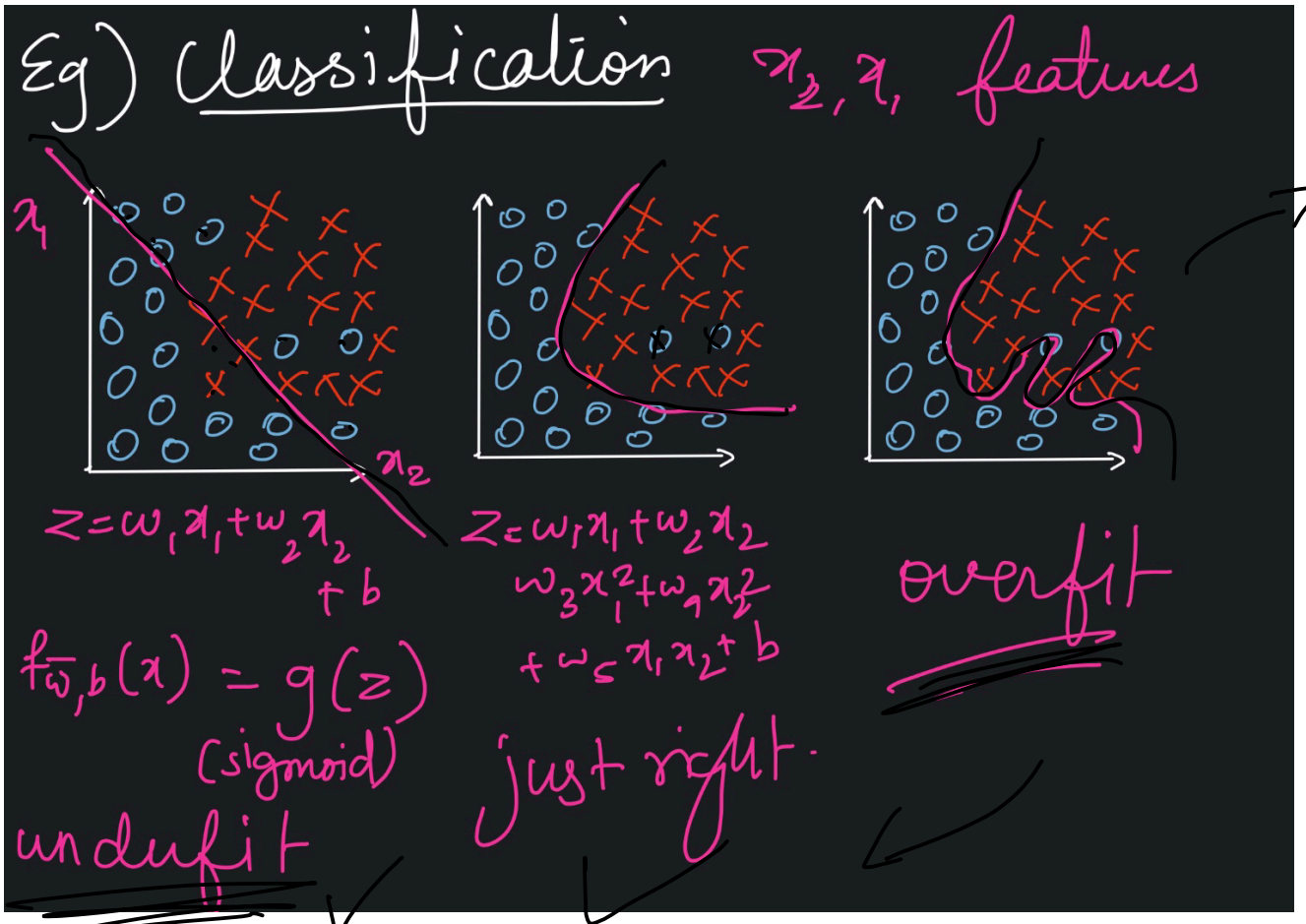
$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

linear reg $f \rightarrow \vec{w} \cdot \vec{x} + b$

logistic reg $t = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$

Overfitting & Vondufitting

↳ high variance ↳ high bias



fix ??

Methods

① Collect more training data → underfit

② Reduce polynomial curves → over

③ Appropriate feature selection → both

④ Regularisation

→ removing feature X

→ reduce the weight given to it ✓

Cost fun

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x_i^{(i)}) - y_i^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

w_j

exploding term.

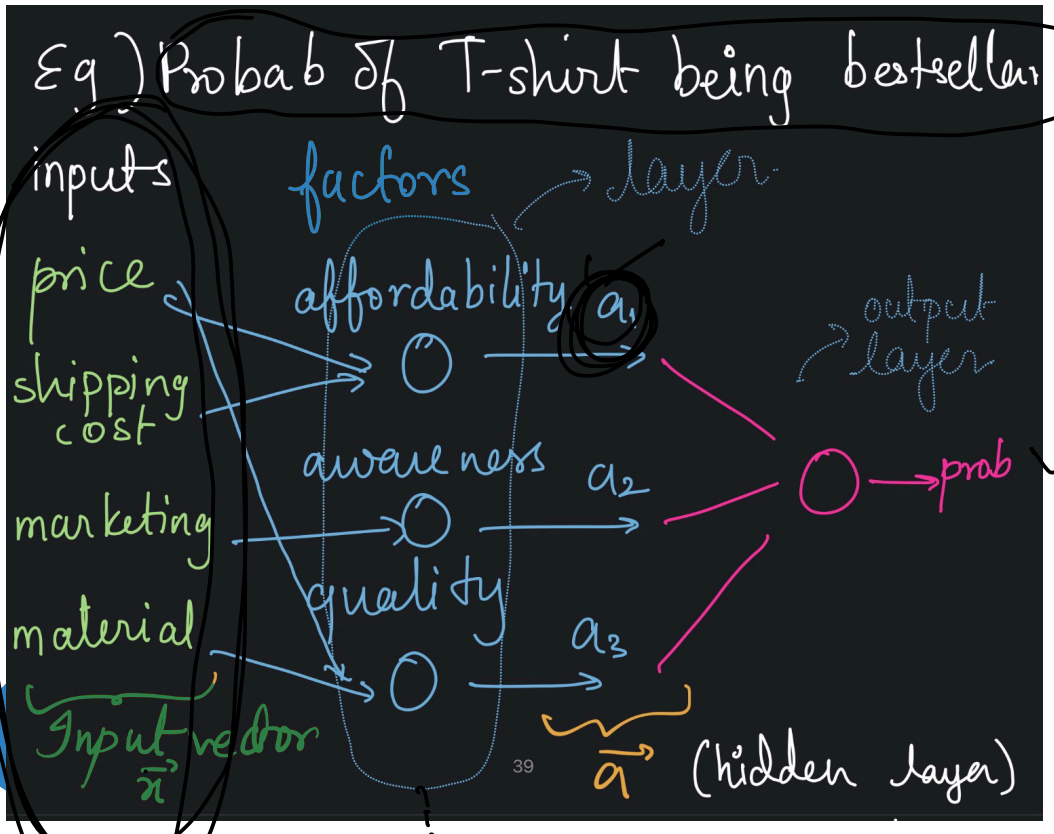
$$w_j^1 = w_j - \alpha \frac{\partial}{\partial w_j} J$$

w_j

w_j, $x_i^{(i)}$ → effect reduced.

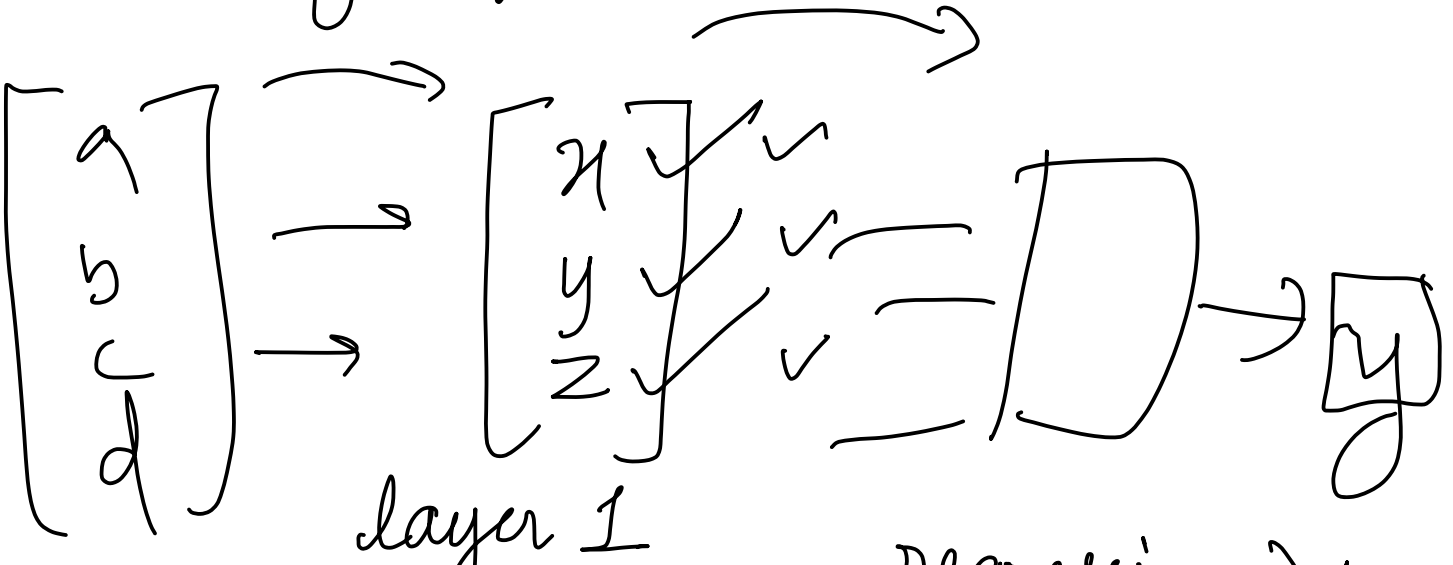
L1, L2 → $\sum w_j$

Neural Network



(layer → intermediate)

Bunch of regression



layer 1

3 neurons

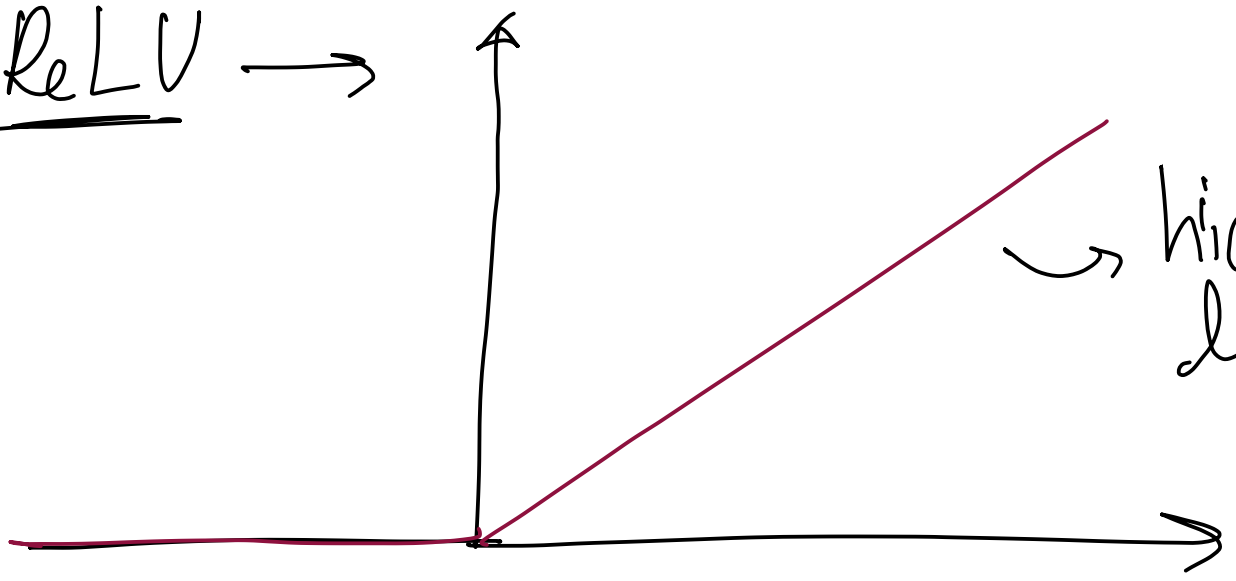
regression

vectors vector

→ Tensorflow ✓

→ layer → activation function

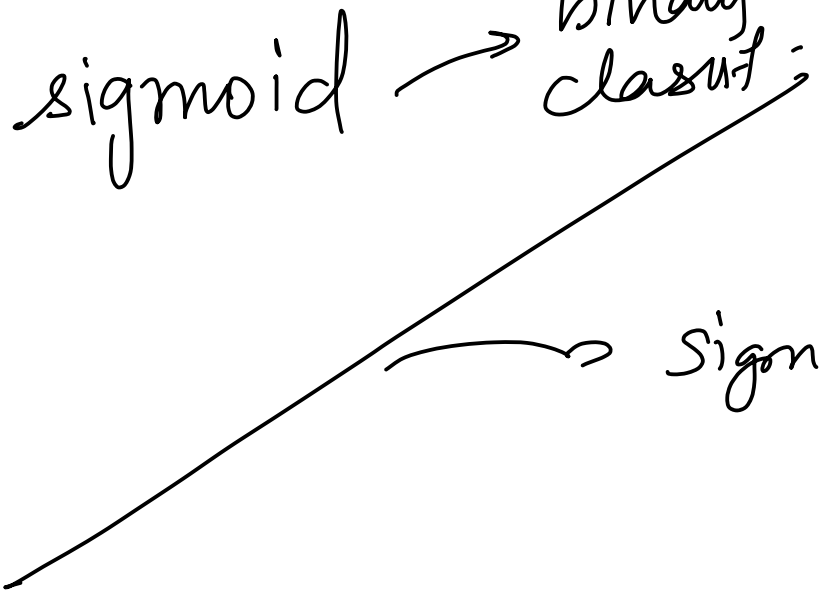
① ReLU →



→ hidden layer

[max (0, x)]

② sigmoid → binary classif:



→ sigmoid ← final layer (prediction) ✓

③ softmax

① More classifications

$z_1 = \bar{w}_1 \cdot \bar{x}_1 + b_1 \rightarrow$ class 1

$\vdots$

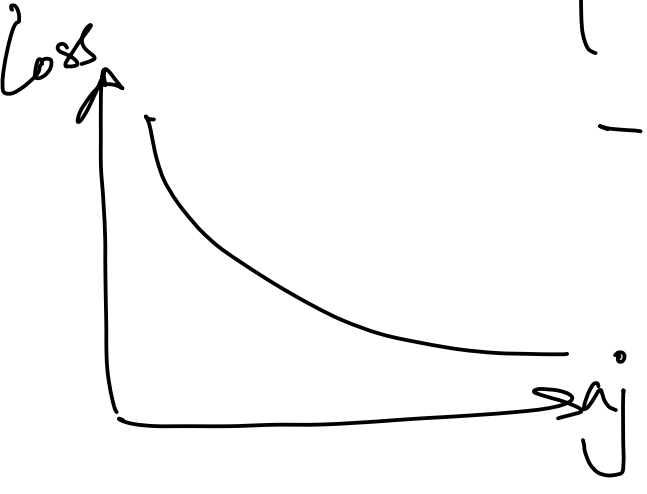
$z_4 = \bar{w}_4 \cdot \bar{x}_4 + b_4 \rightarrow$ class 4

$$a_1 = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3} + e^{z_4}}$$

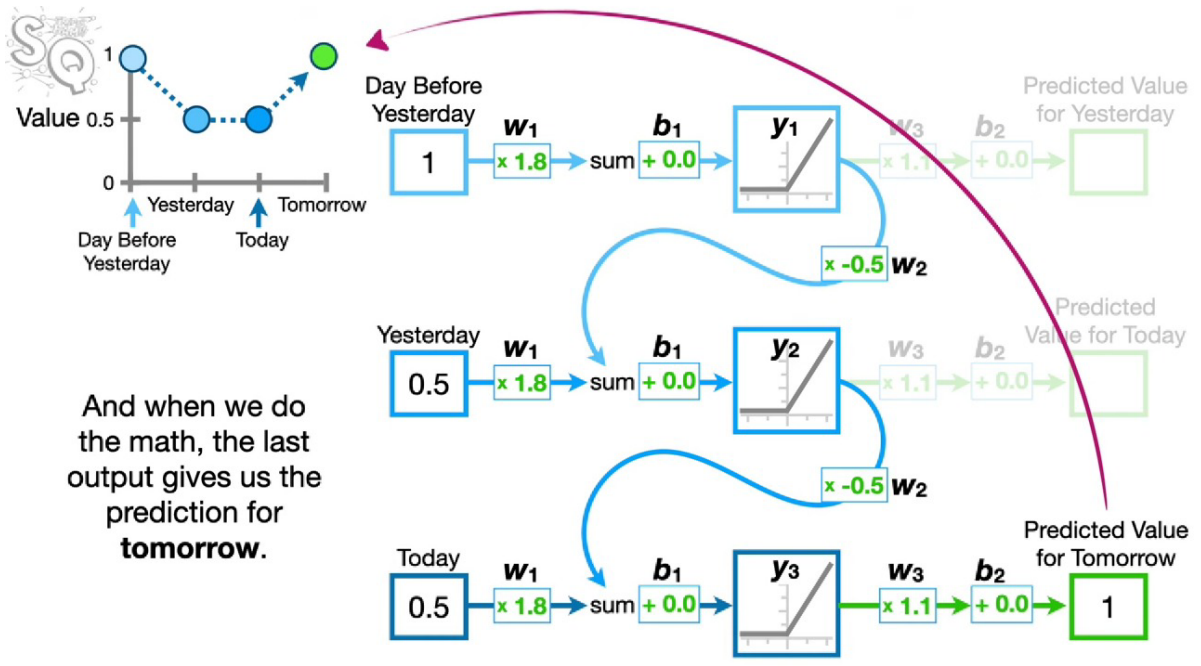
$$a_j = \frac{e^{z_j}}{\sum e^{z_k}} \rightarrow \text{softmax layer}$$

Cost →

$$\text{loss}(a_1, \cdots a_N, y) = \begin{cases} -\log a_1 & y = 1 \\ -\log a_2 & y = 2 \\ \vdots & \\ -\log a_n & y = N \end{cases}$$



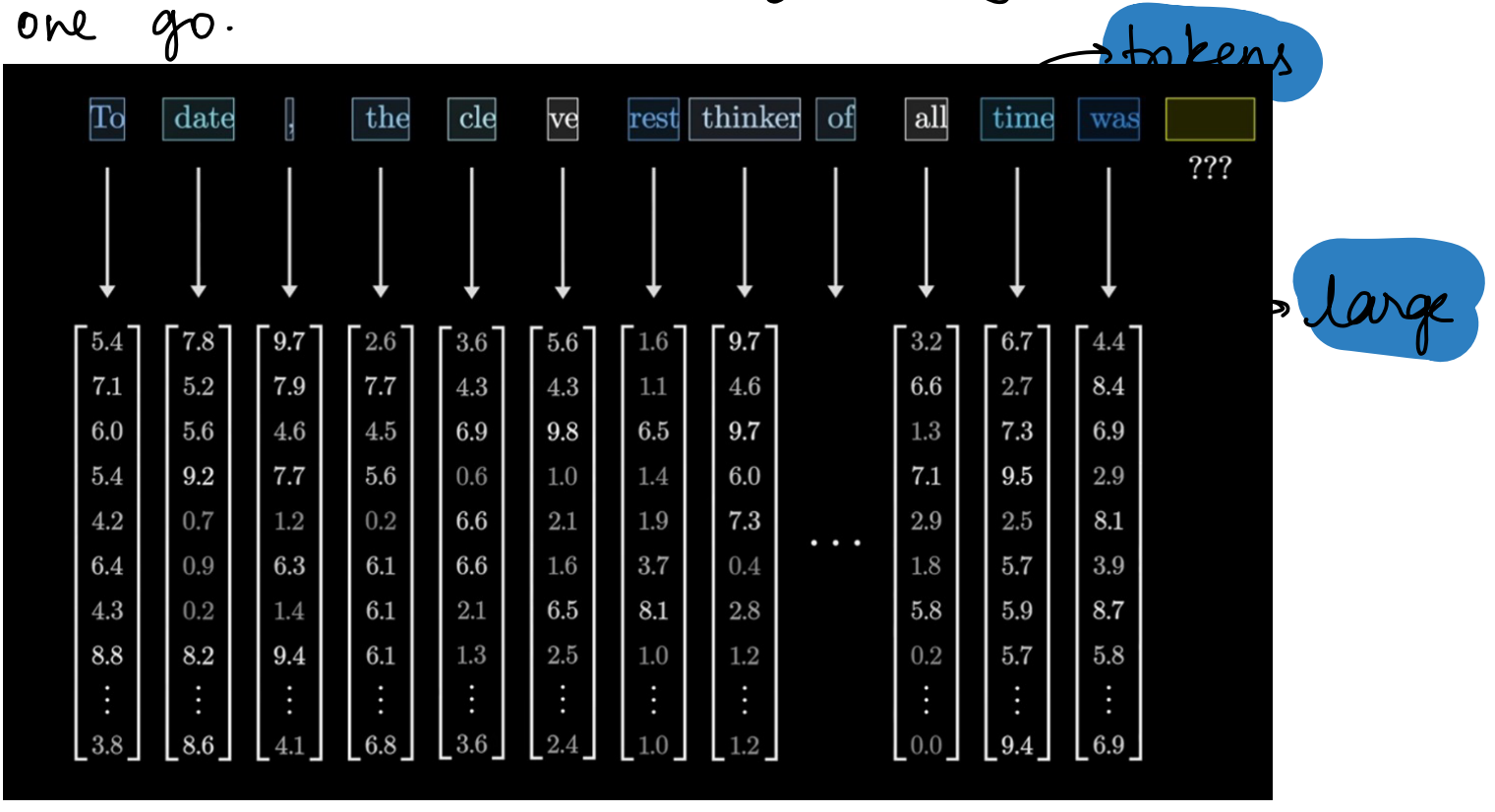
RNNs



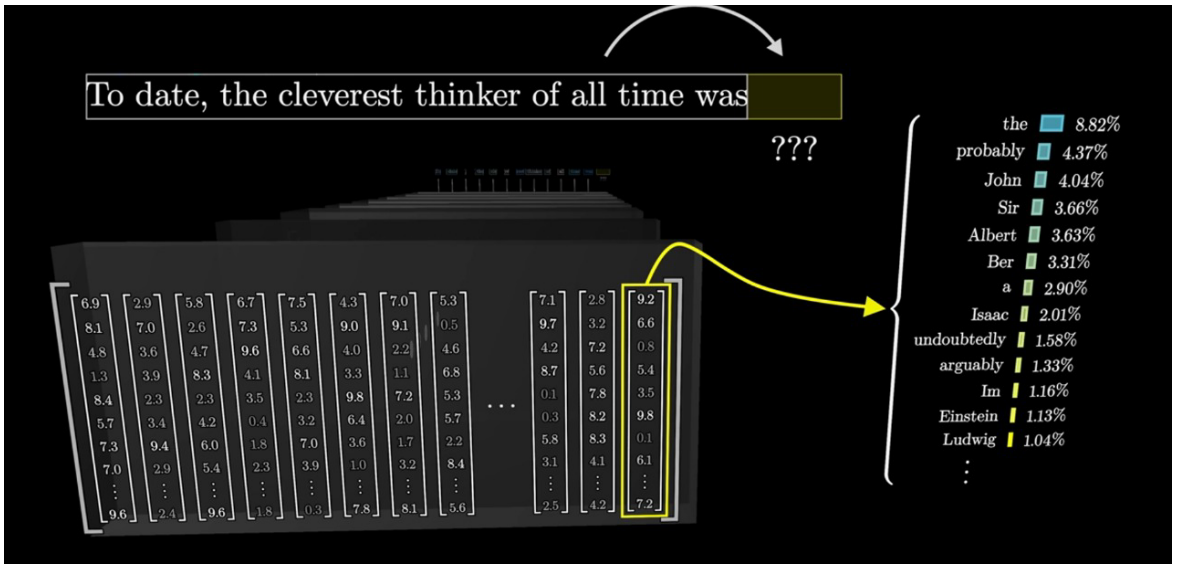
Sequential data, customisable no. of inp.

LLMs (Attention mechanism)

→ Predict next word by taking all words in one go.

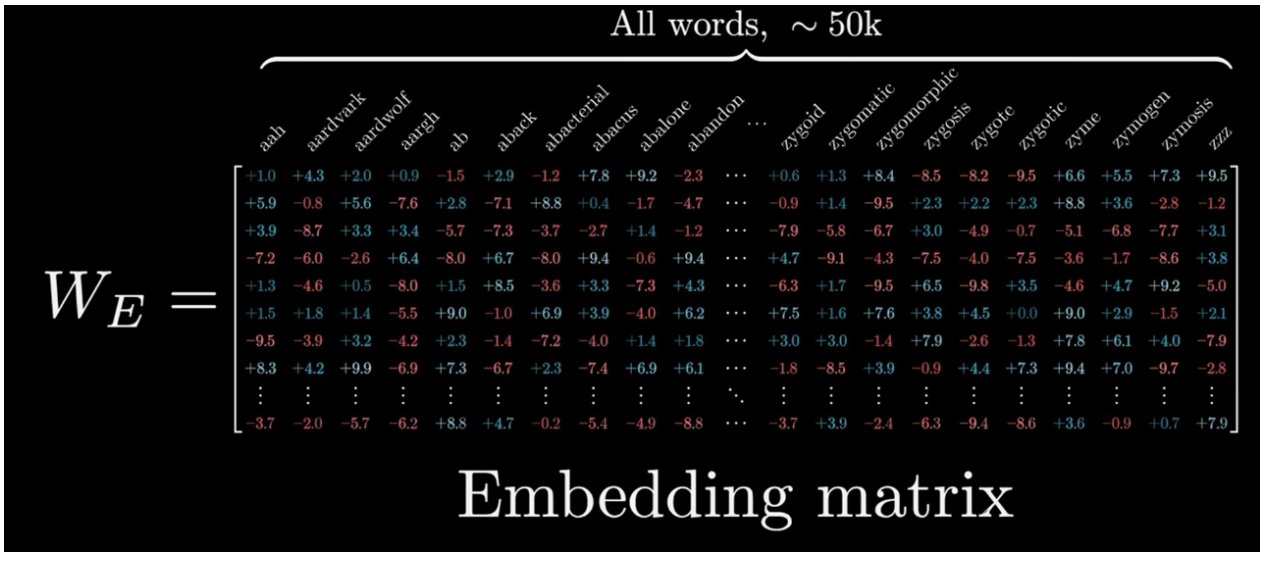


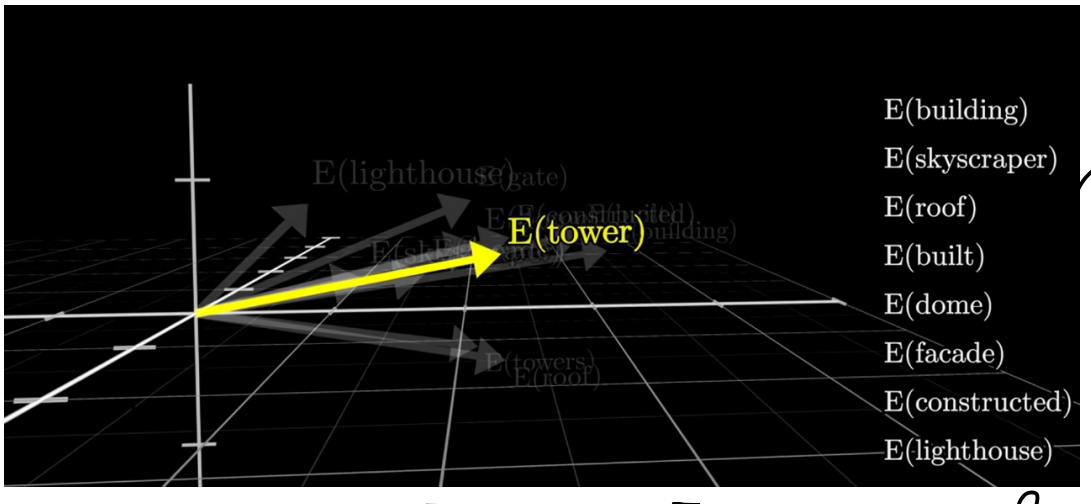
A machine learning model fashion model



semantic meaning

→ probability of words





semantic

$\overrightarrow{\text{vectors}} = \overrightarrow{\text{word}_1} - \overrightarrow{\text{word}_2}$ → dot & cross product (distance) for 'Attention' mechanism

→ content size