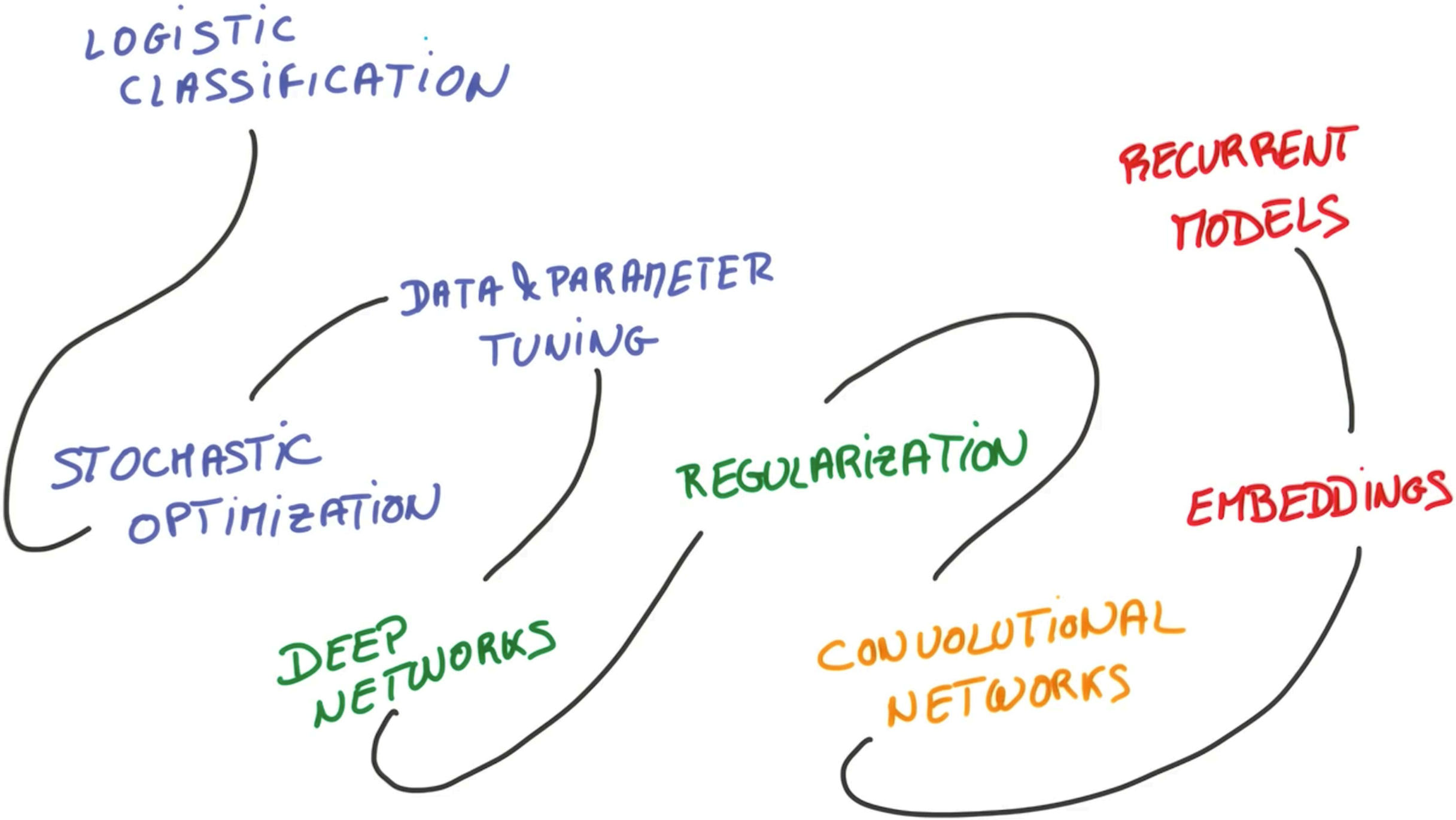
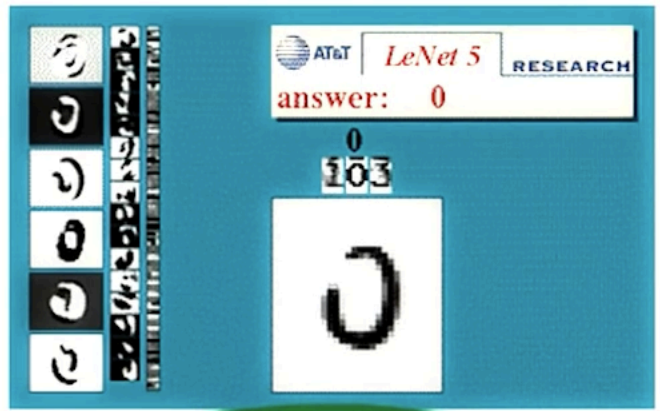
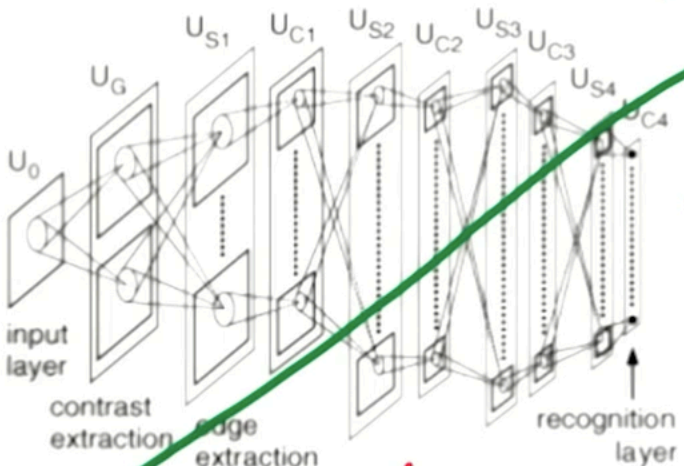

LOGISTIC CLASSIFICATION
STOCHASTIC OPTIMIZATION
DATA & PARAMETER TUNING
DEEP NETWORKS
REGULARIZATION
CONVOLUTIONAL NETWORKS
RECURRENT MODELS
EMBEDDINGS

NEURAL NETWORKS

LE CUN'S LENET-5

FUKUSHIMA'S NEOCOGNITRON





1980's

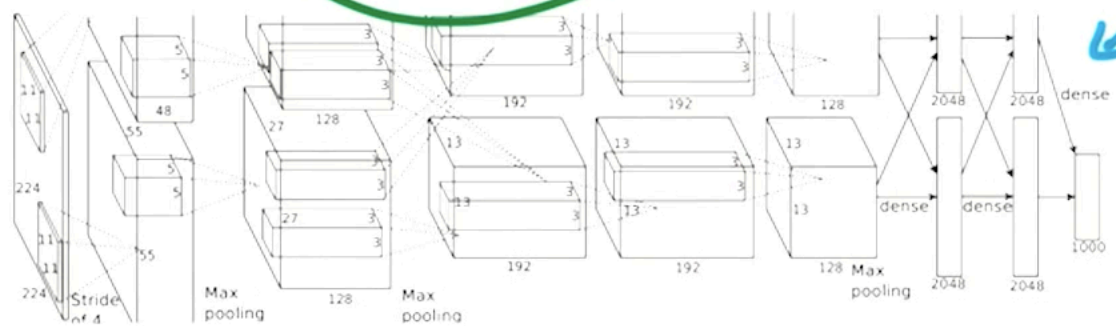
1990's

2000's

2010's

...

KRIZHEVSKY'S ALEXNET



NEURAL NETWORKS ⟶ DEEP LEARNING

- 2009 : SPEECH RECOGNITION
- 2012 : COMPUTER VISION
- 2014 : MACHINE TRANSLATION

THANK YOU GAMERS!!!

(1) DATA!

(2) GPUs!

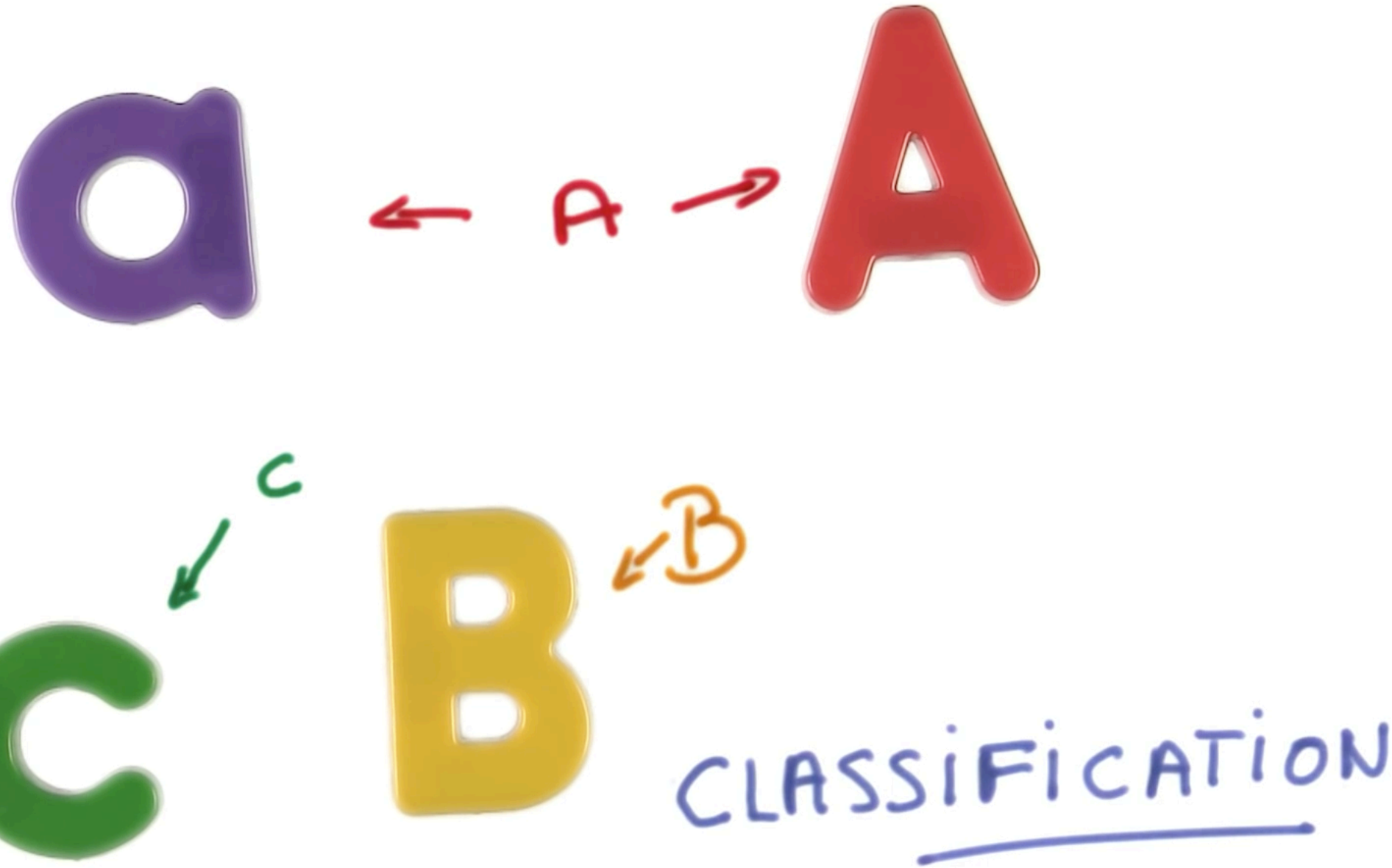
REGRESSION

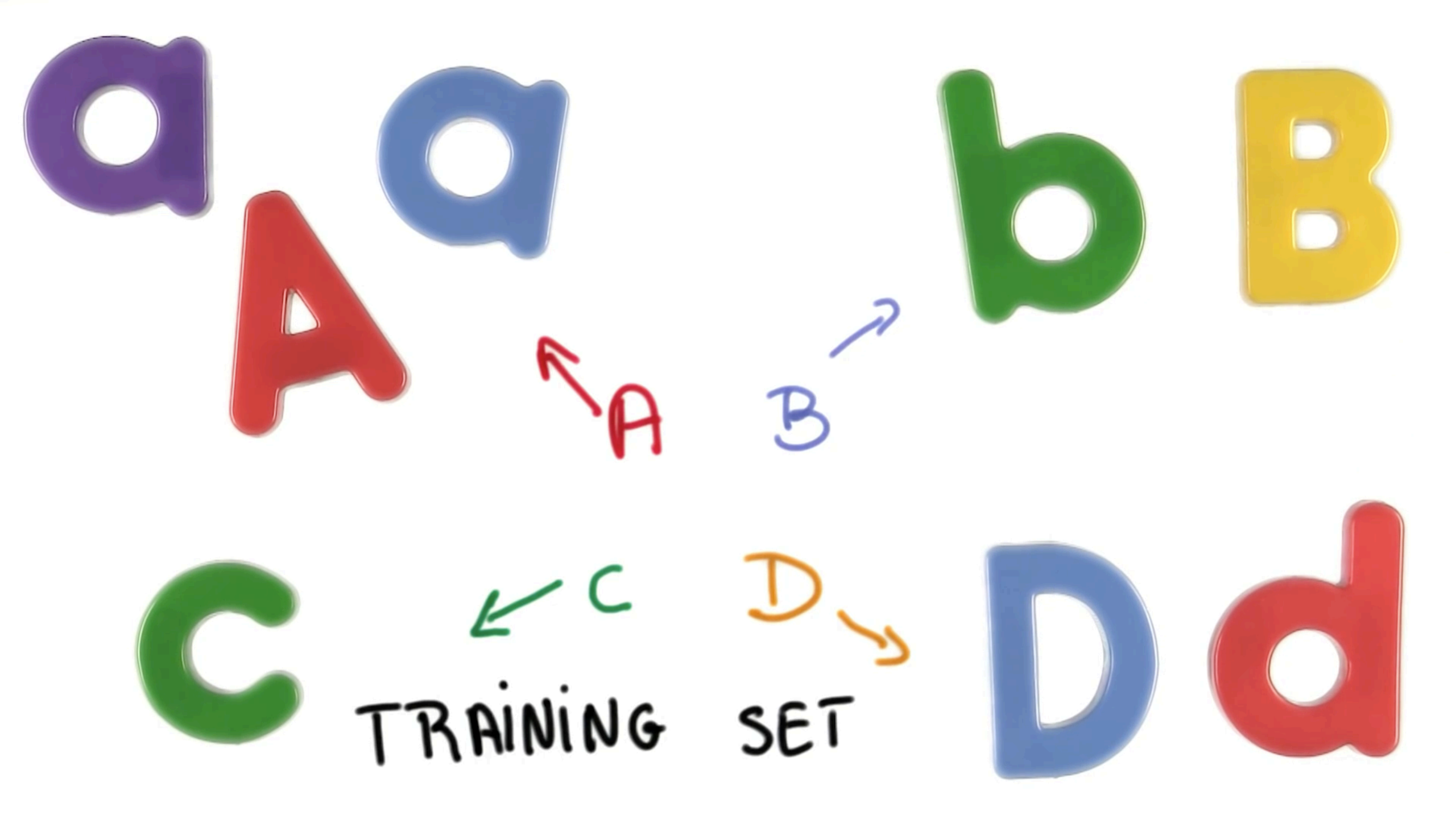
REINFORCEMENT LEARNING

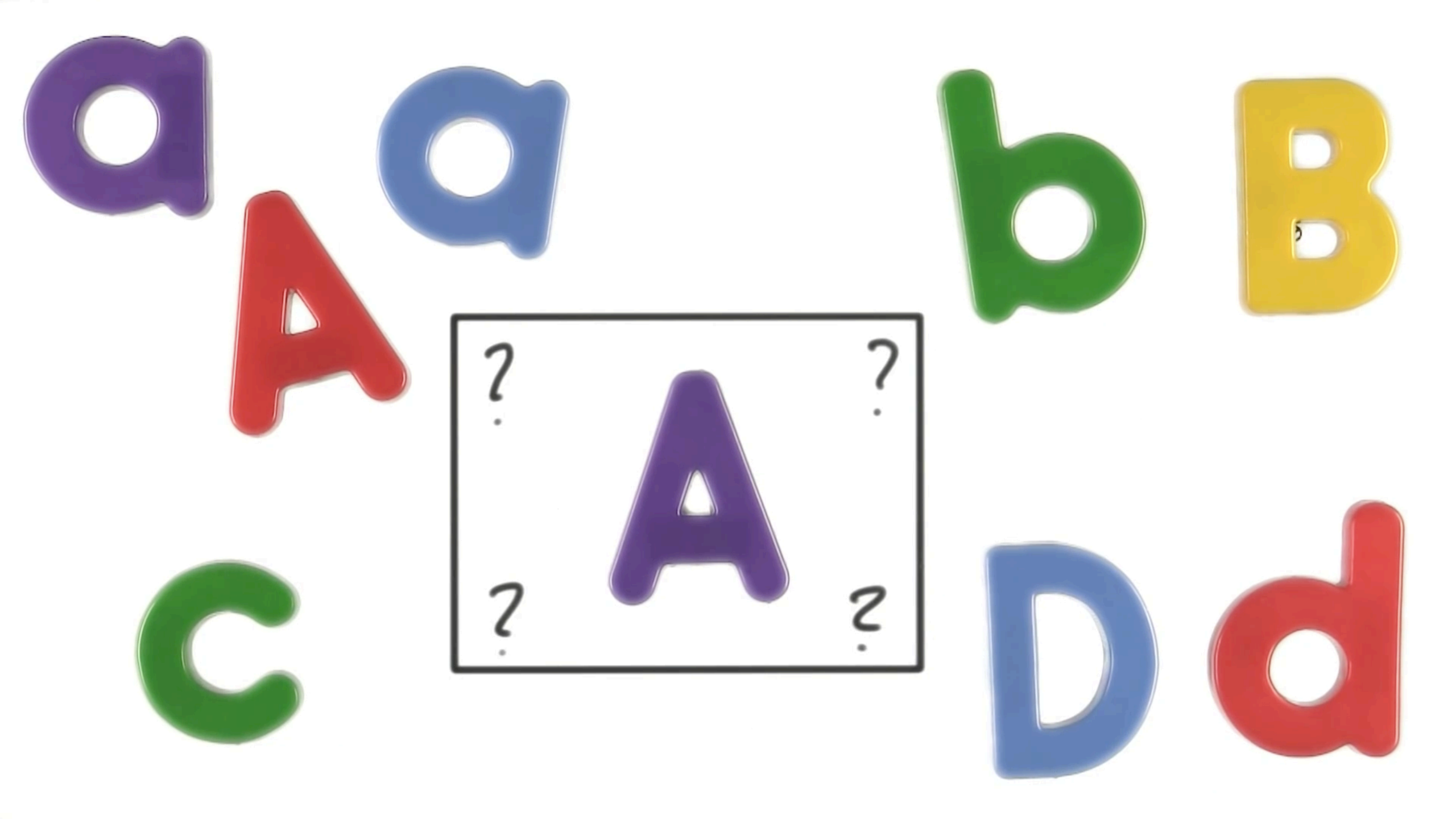
CLASSIFICATION

RANKING

DETECTION

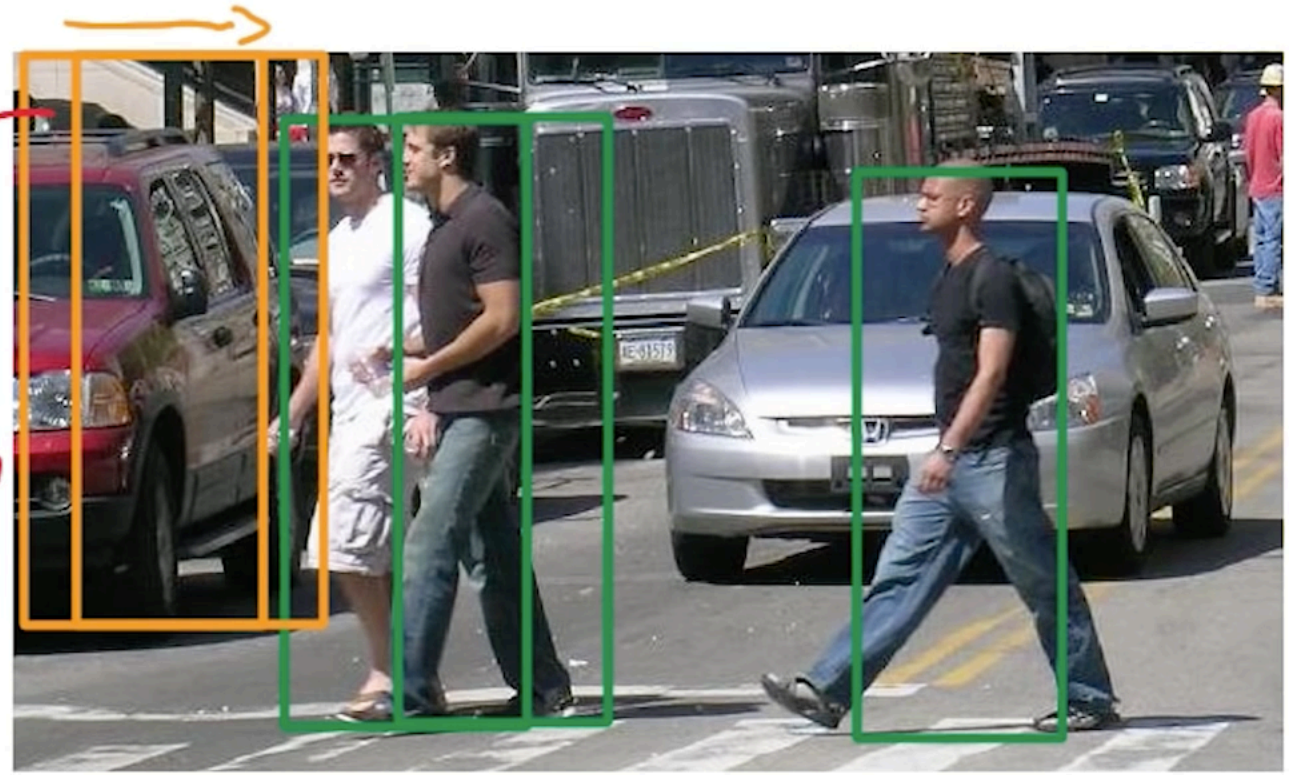

a
A
A
C
c
B
B
CLASSIFICATION


A
B
C
D
TRAINING SET


a
a
A
b
B
?
?
A
?
?
C
D
d

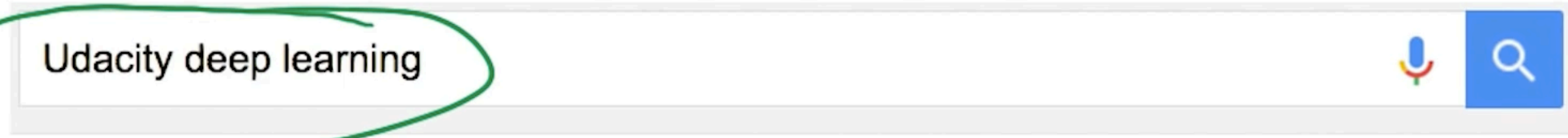
DETECTION

PEDESTRIAN
VS.
NO PEDESTRIAN



Use a binary pedestrian / no pedestrian classifier. Slide it over all possible locations in the image.

RANKING



Web News Videos Images Shopping More Search tools

About 46,700 results (0.64 seconds)

Machine Learning: Supervised Learning - Udacity
https://www.udacity.com/wiki/ud675 Udacity
Oct 3, 2014 - Ethem Alpaydın, Introduction to Machine **Learning**. Second Edition. ...

CLASSIFIER

↓

RELEVANT!

Use a classifier that takes the pair (query, web page) as an input, and classifies the pair as relevant / not relevant

LOGISTIC CLASSIFIER

$$WX + b = Y$$

LOGISTIC CLASSIFIER

$$WX + b = Y$$

WEIGHTS

BIAS

TRAINED

LOGISTIC CLASSIFIER

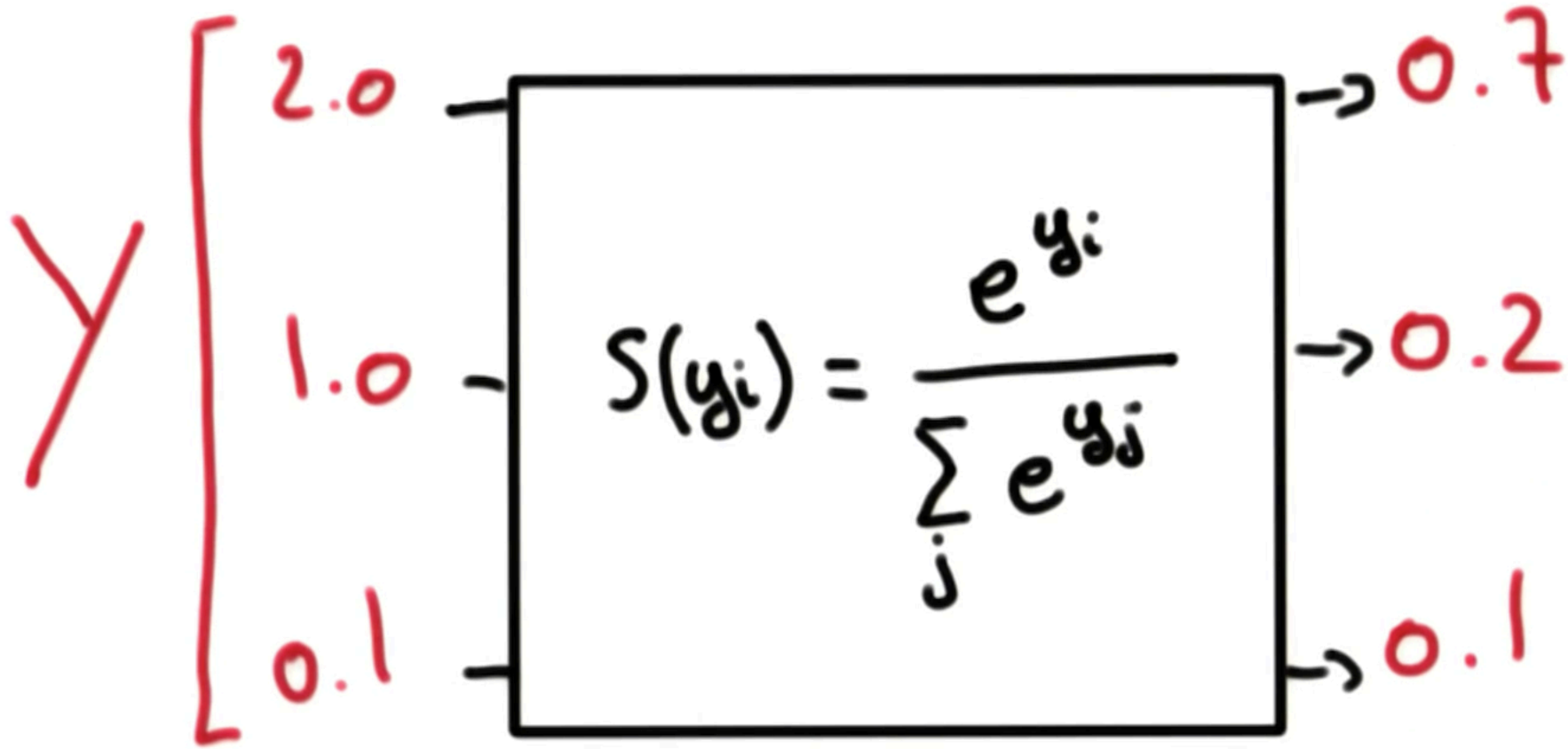
$$WX + b = Y$$

Y = 2.0 → $p = 0.7$

Y = 1.0 → $p = 0.2$

Y = 0.1 → $p = 0.1$

SOFTMAX



'LOGITS'

SCORES ⟷ PROBABILITIES

```
"""Softmax."""

scores = [3.0, 1.0, 0.2]

import numpy as np

def softmax(x):
    """Compute softmax values for x."""
    return np.exp(x) / np.sum(np.exp(x), axis=0)

print(softmax(scores))

# Plot softmax curves
import matplotlib.pyplot as plt
x = np.arange(-2.0, 6.0, 0.1)
scores = np.vstack([x, np.ones_like(x), 0.2 * np.ones_like(x)])

plt.plot(x, softmax(scores).T, linewidth=2)
plt.show()

```

Reset | Test Run | Submit

Get Help

Discussion Topics

Coaches Lounge

Report an Issue

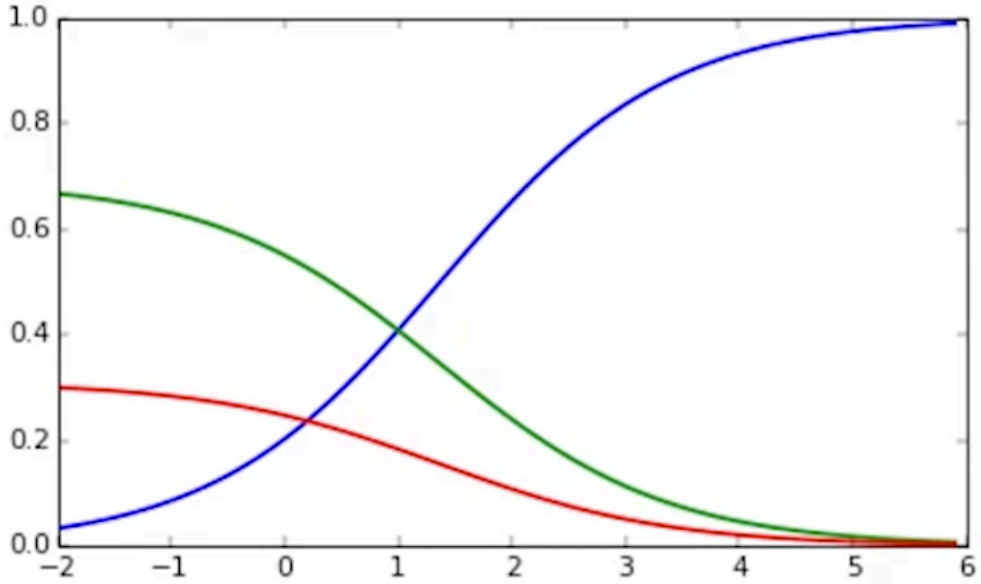
Instructor Notes

Downloadables

There are no relevant downloads for this part of

Reset | Test Run | Submit

```
[ 0.8360188   0.11314284  0.05083836]
```



Instructor Notes

Downloadables

There are no relevant downloads for this part of the course.

```
13
14
15
16
17
18
19
20 # Multiply the scores by 10. What happens?
21 scores = np.array([3.0, 1.0, 0.2])
22 print(softmax(scores * 10))
23 print(softmax(scores / 10))
24
25
26
27
28
29
30
31
32
33
```

Reset | Test Run | Submit

```
[ 0.8360188   0.11314284  0.05083836]
```

Get Help

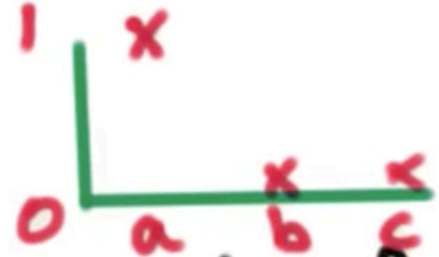
Discussion Topics

Coaches Lounge

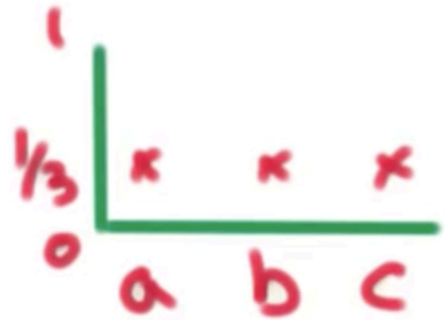
Report an Issue

x10

- ✓ probabilities get close to either 0.0 or 1.0



- ○ probabilities get close to the uniform distribution



```
13
14
15
16
17
18
19
20 # Multiply the scores by 10. What happens?
21 scores = np.array([3.0, 1.0, 0.2])
22 print(softmax(scores * 10))
23 print(softmax(scores / 10))
24
25
26
27
28
29
30
31
32
33
```

Reset | Test Run | Submit

```
[ 0.8360188   0.11314284  0.05083836]
```

Get Help

Discussion Topics

Coaches Lounge

Report an Issue

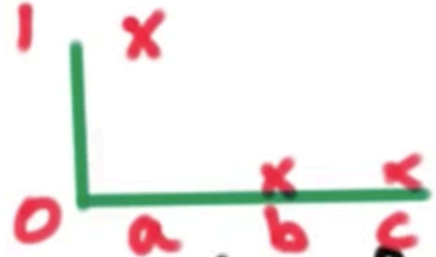
Get Help

Discussion Topics

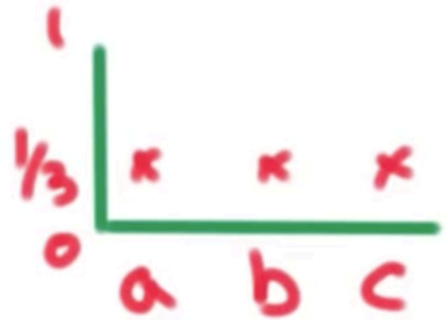
Report an Issue

÷10

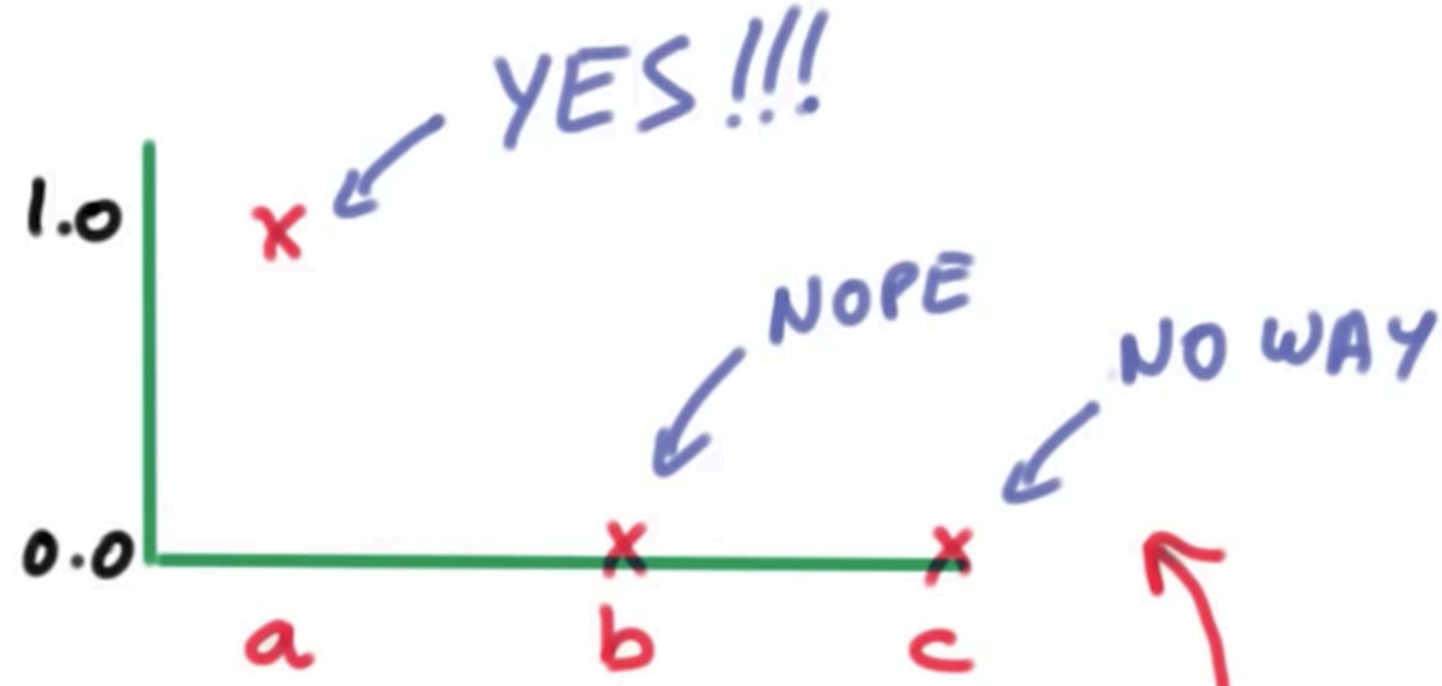
○ probabilities get close to either 0.0 or 1.0



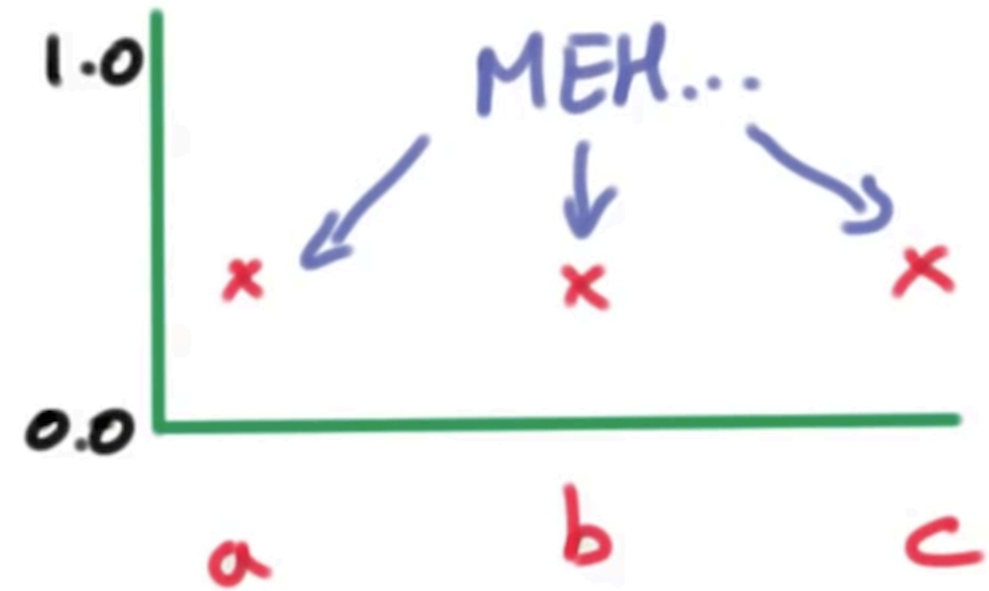
✓ probabilities get close to the uniform distribution

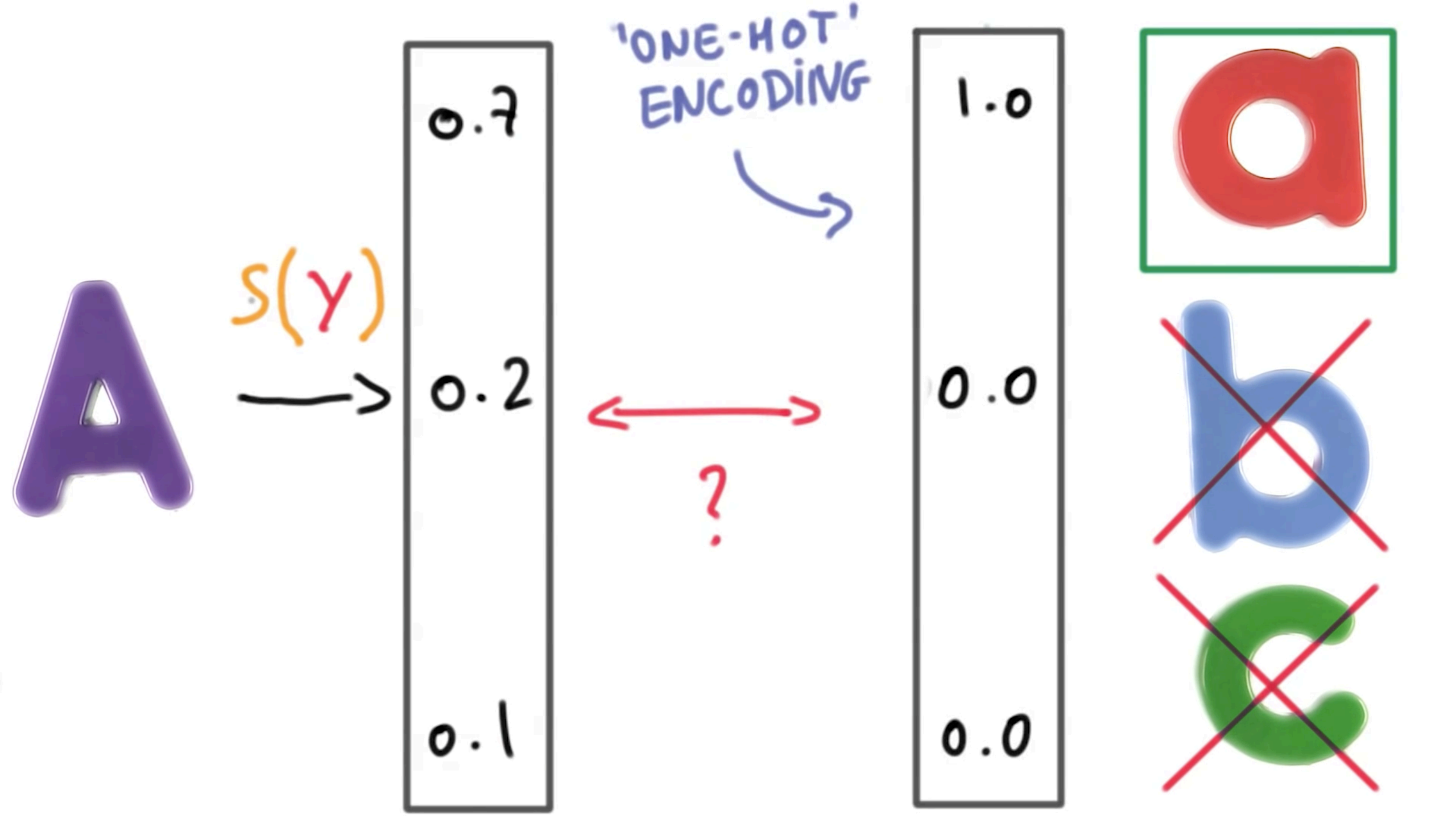


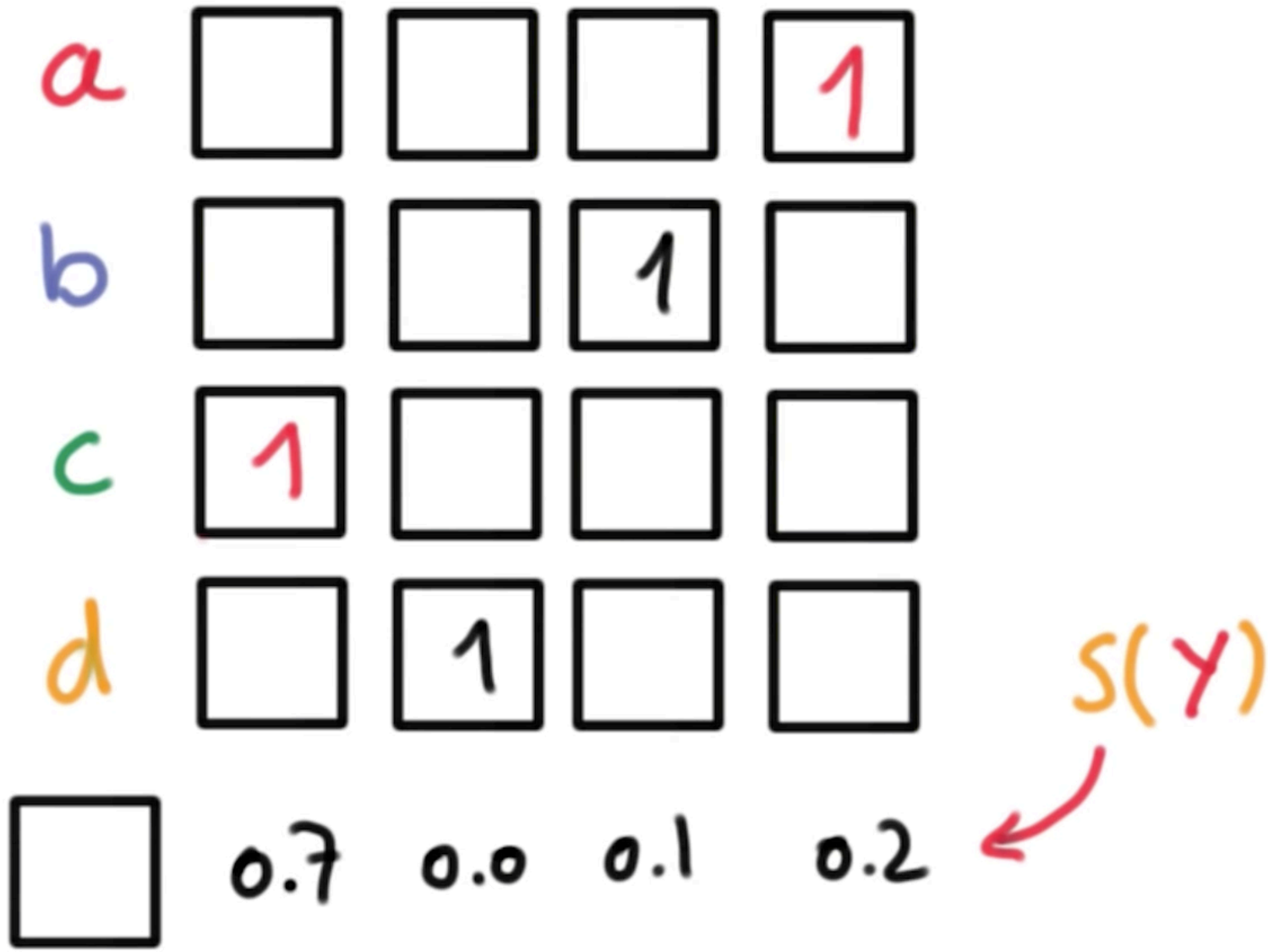
$S(Y \times 10)$

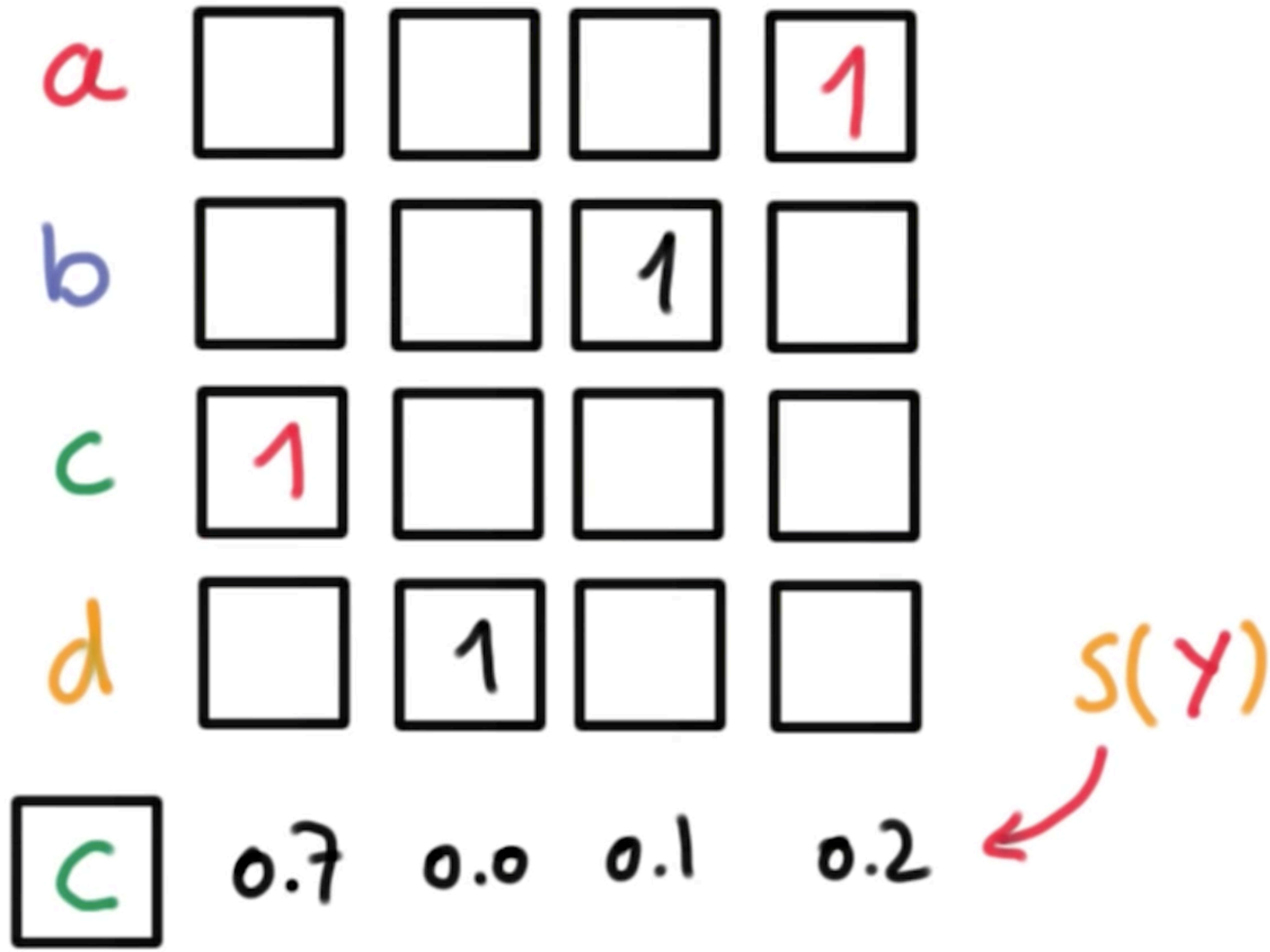


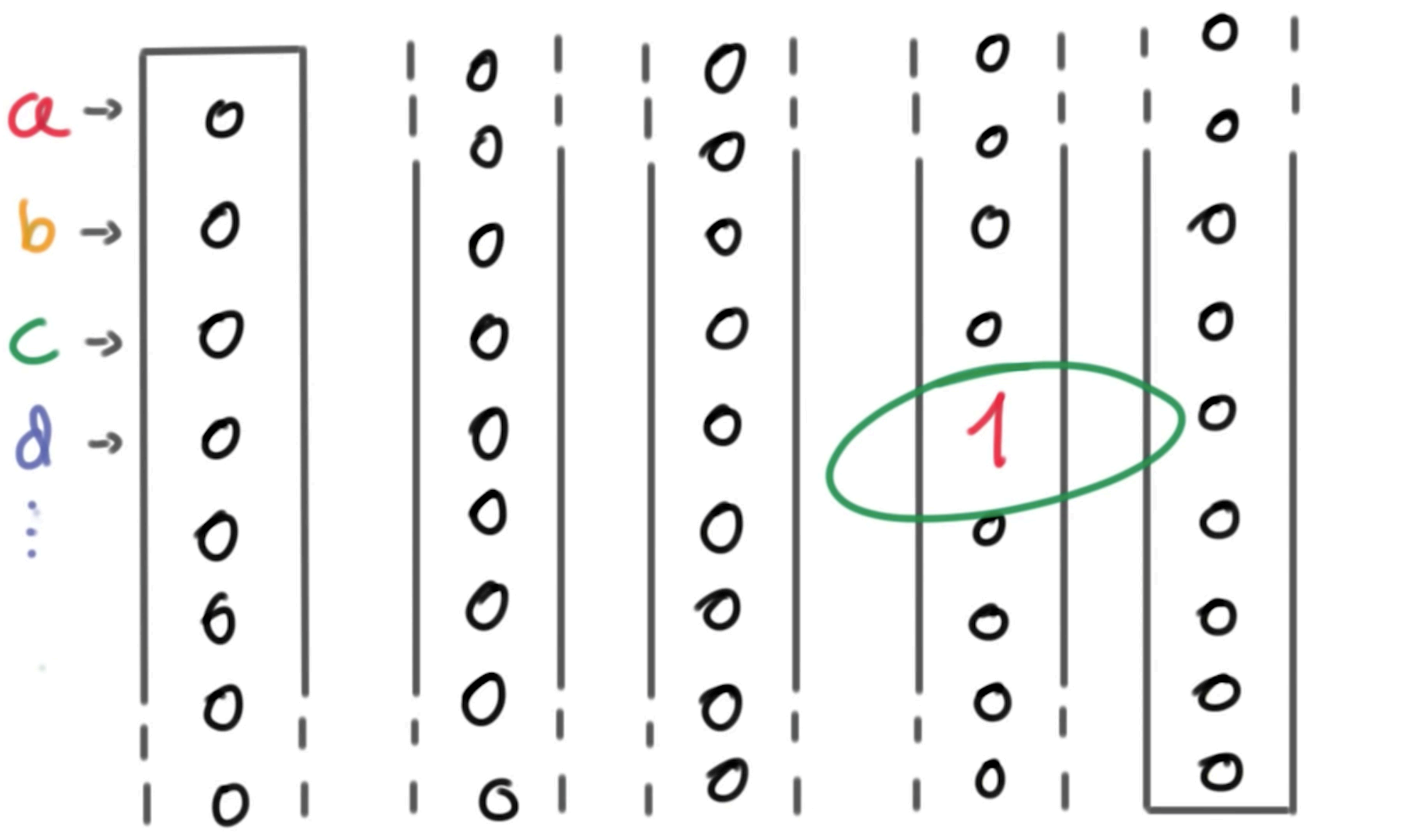
$S(Y / 10)$

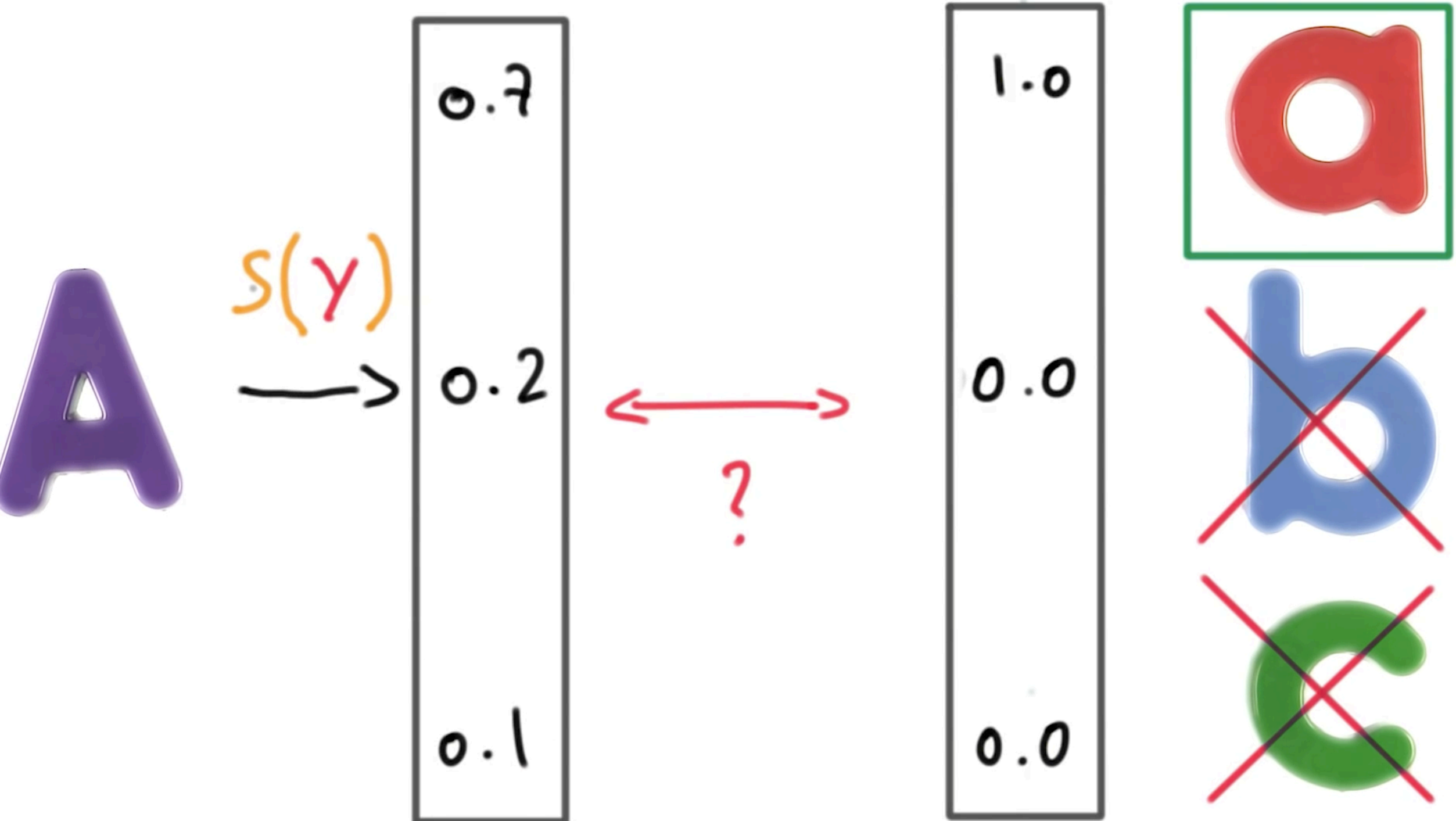



'ONE-HOT' ENCODING
A
S(Y)
0.7
0.2
0.1
?
1.0
0.0
0.0

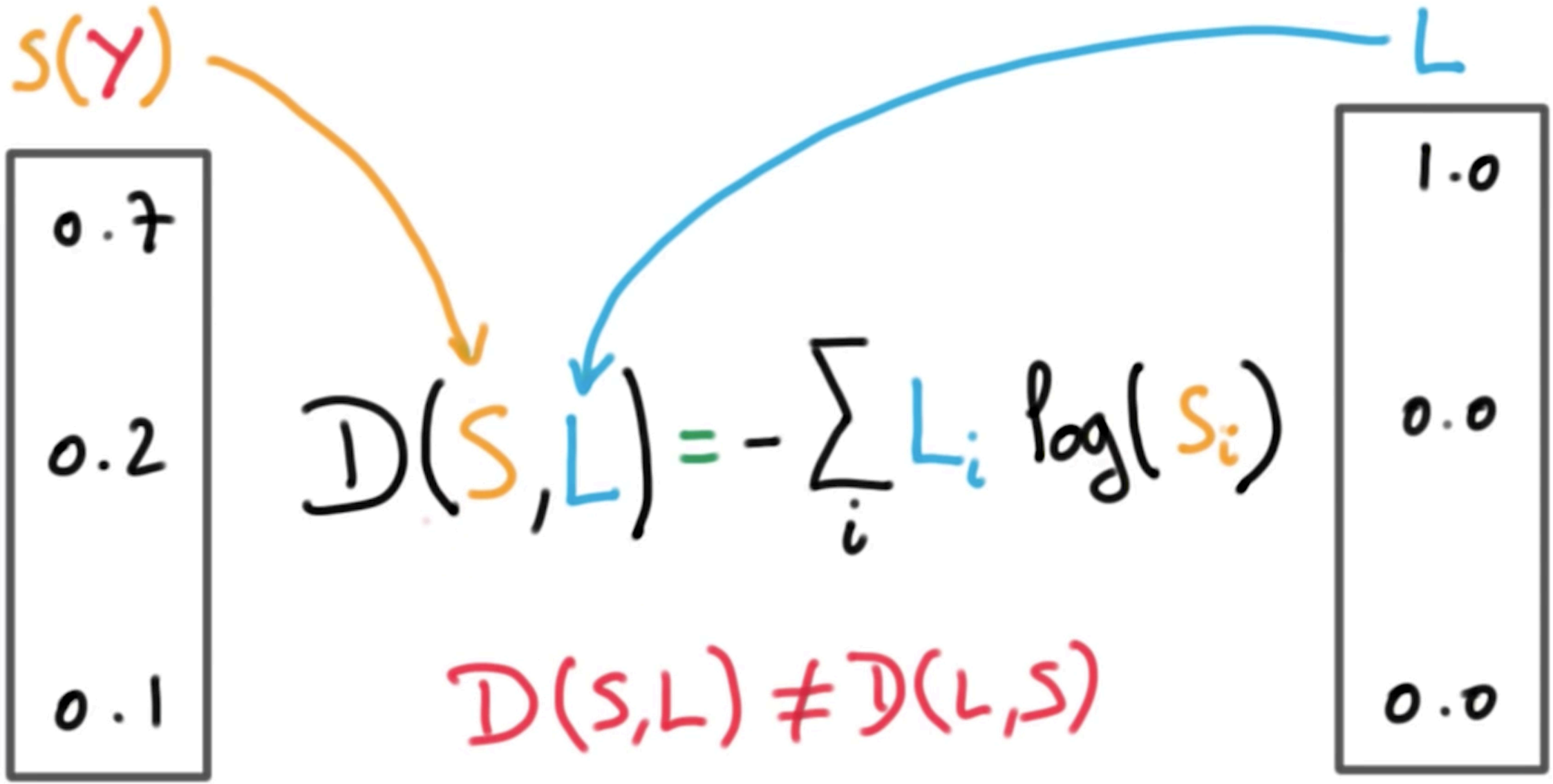

a
1
b
1
c
1
d
1
0.7
0.0
0.1
0.2
S(Y)


a
1
b
1
c
1
d
1
S(Y)
c
0.7
0.0
0.1
0.2

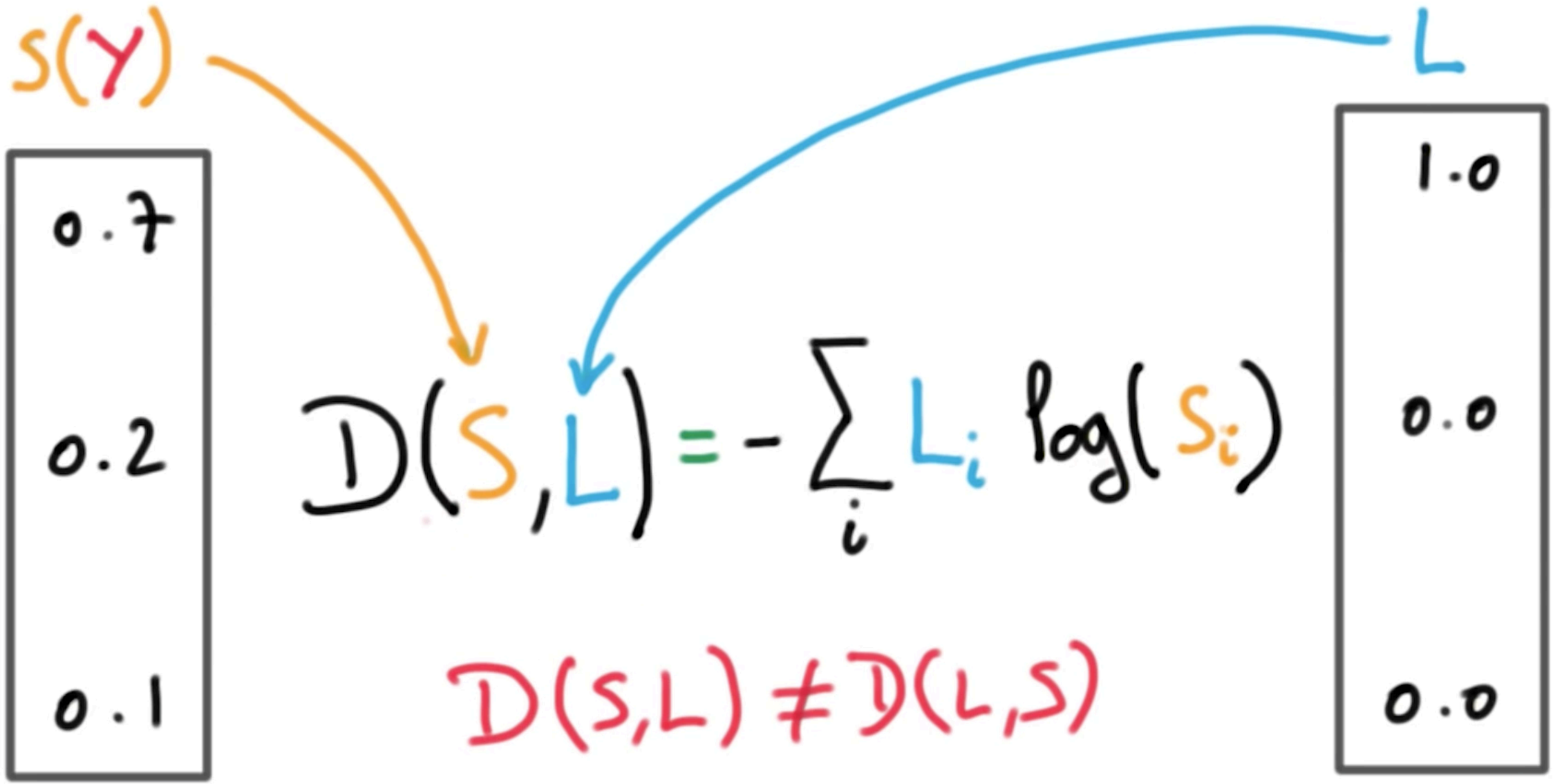

a
b
c
d
1

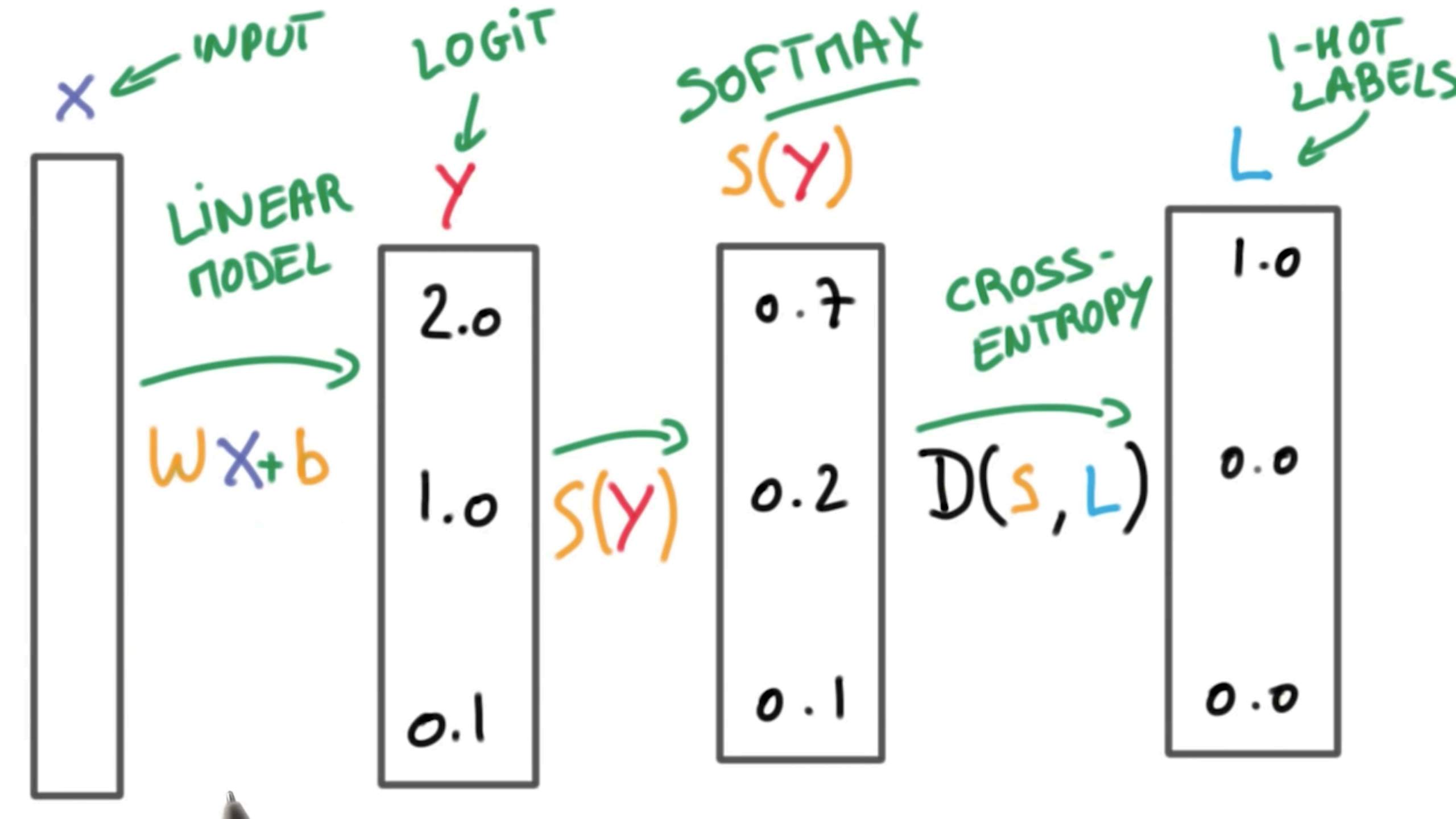

A
S(y)
0.7
0.2
0.1
?
1.0
0.0
0.0

CROSS-ENTROPY



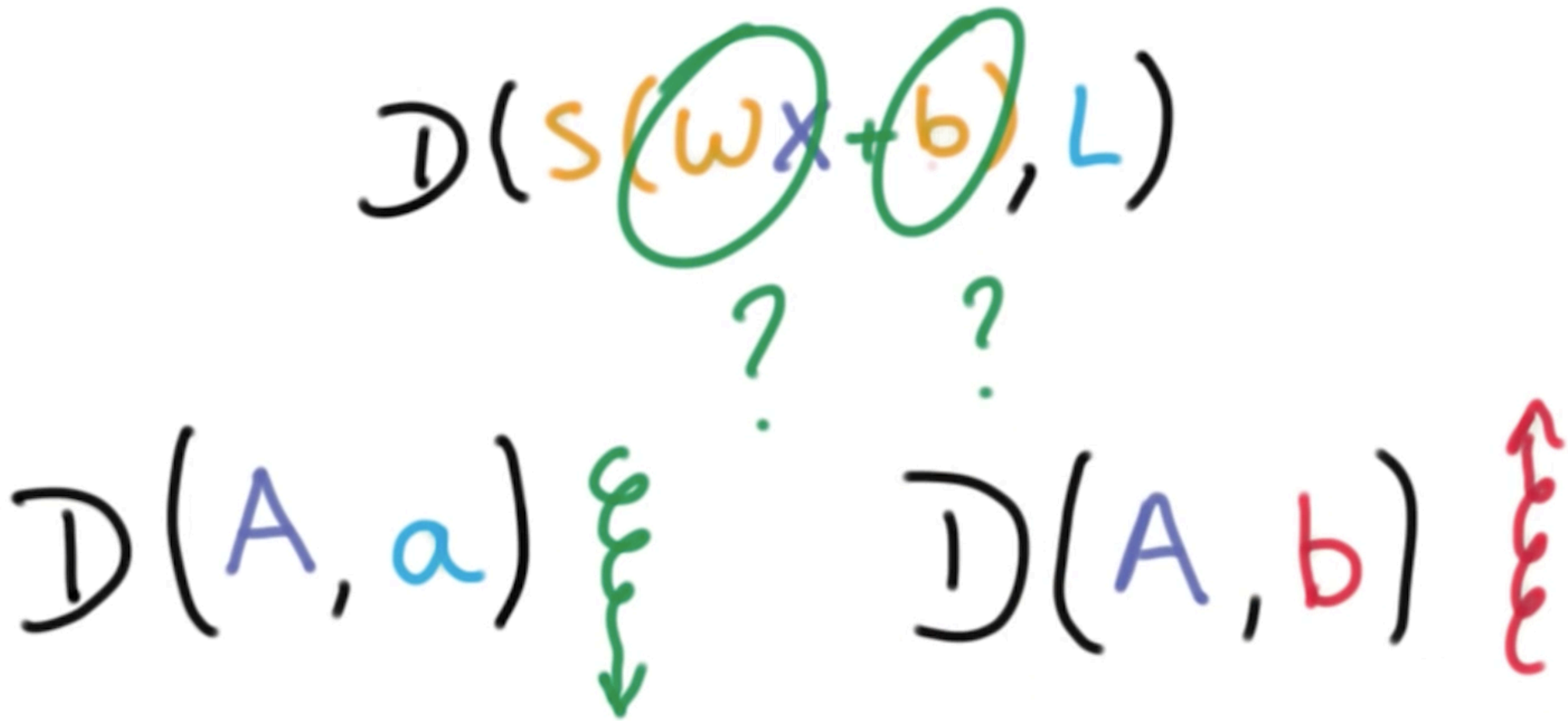
CROSS-ENTROPY

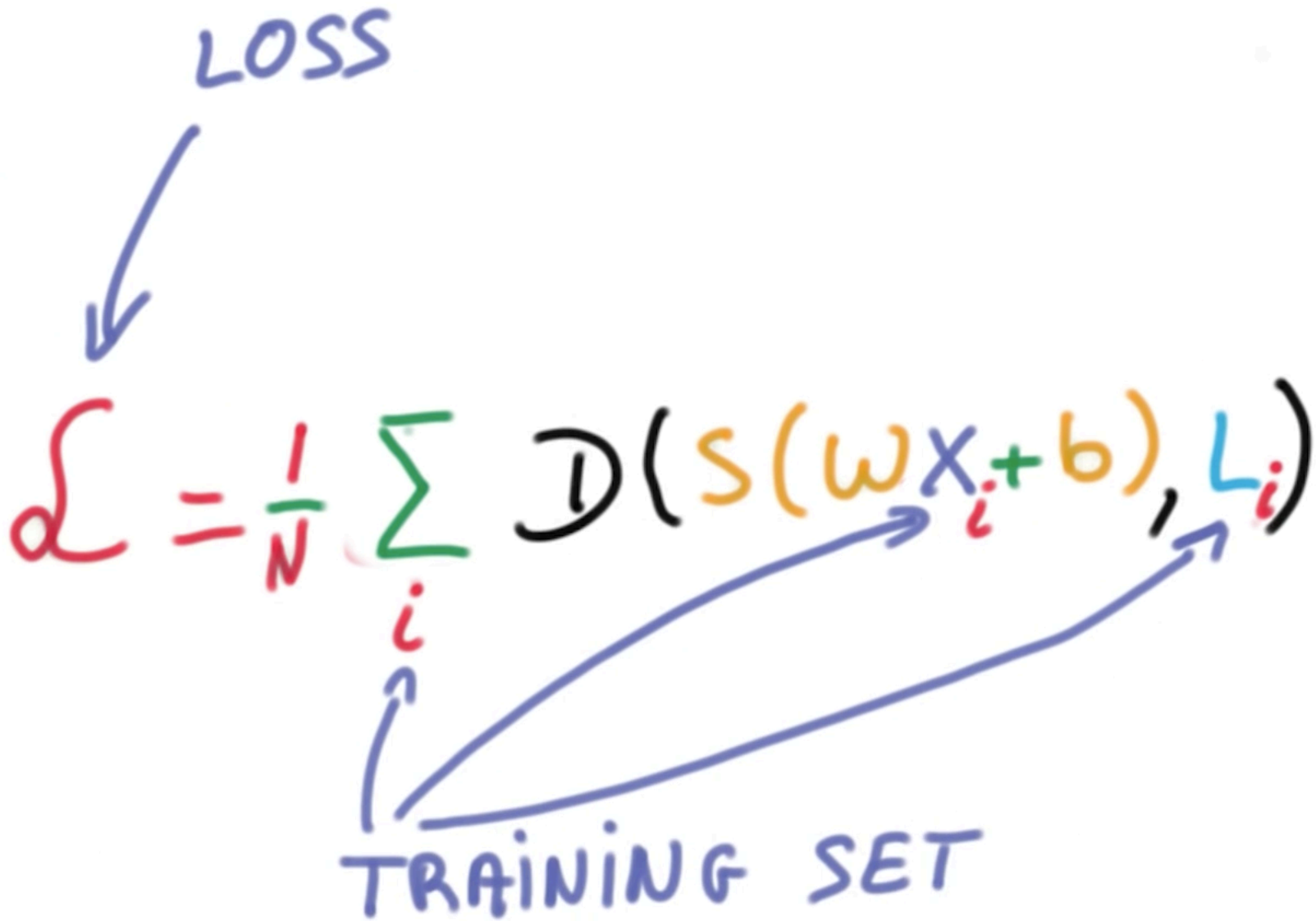



X
INPUT
LINEAR MODEL
WX+b
Y
LOGIT
2.0
1.0
0.1
S(Y)
S(Y)
SOFTMAX
0.7
0.2
0.1
D(S, L)
CROSS-ENTROPY
L
1-HOT LABELS
1.0
0.0
0.0

MULTINOMIAL LOGISTIC CLASSIFICATION

$$D(S(WX+b), L)$$


D(s(wx+b), L)
?
?
D(A, a)
D(A, b)


LOSS
$\mathcal{L} = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$
i
TRAINING SET

LOSS = AVERAGE CROSS-ENTROPY

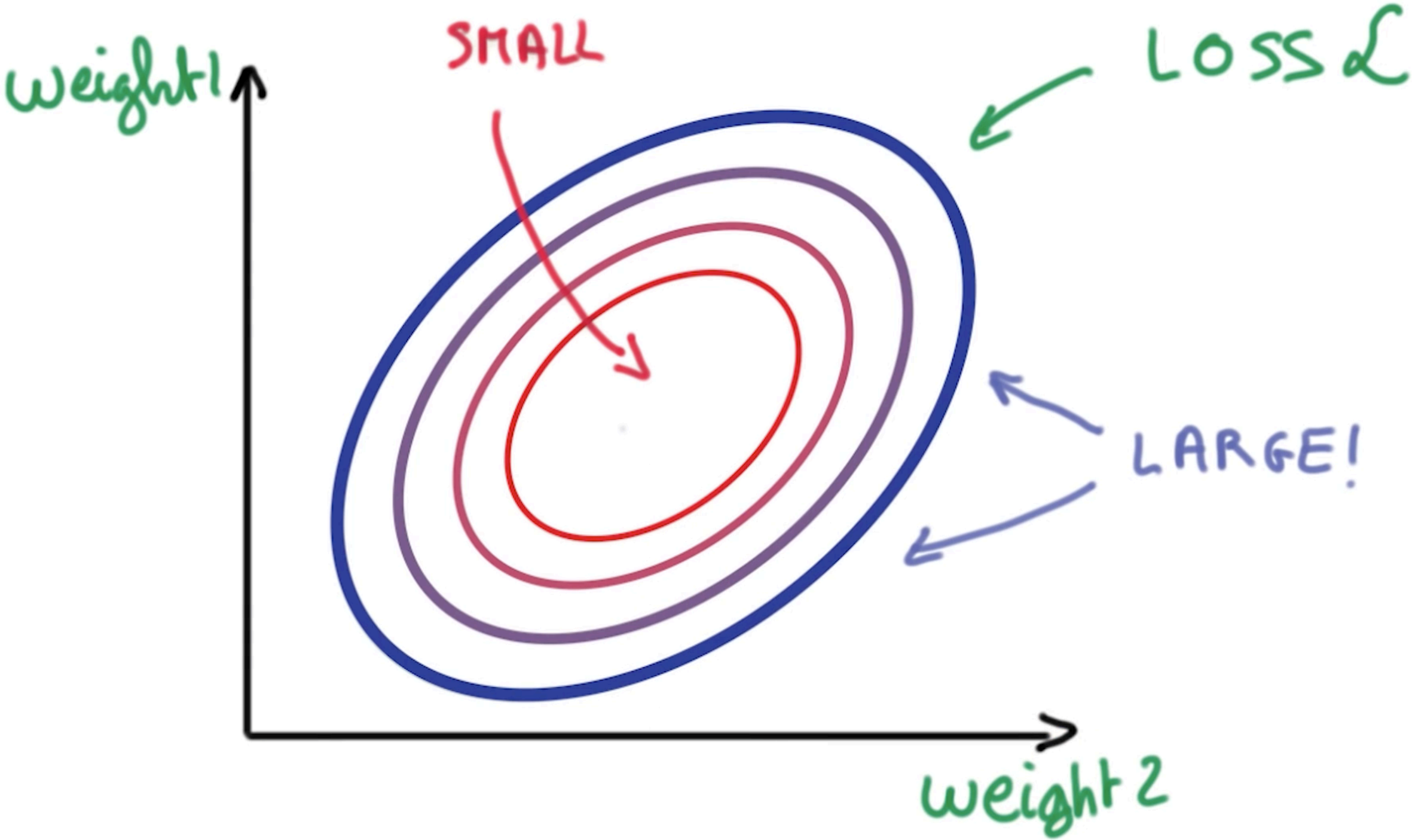
BIG MATRIX!!

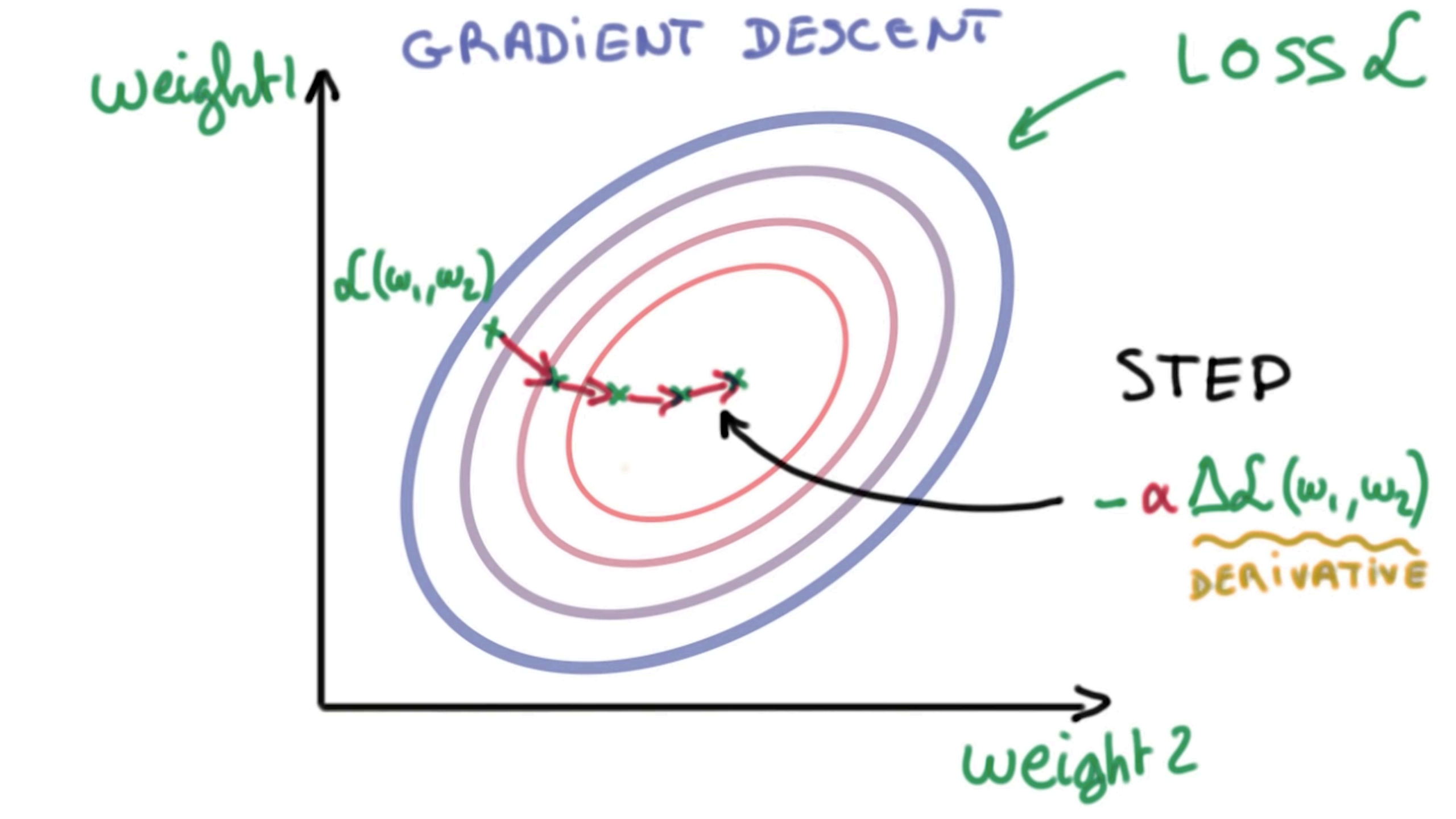
$$\mathcal{L} = \frac{1}{N} \sum_{i} D(S(WX_i + b), L_i)$$

BIG SUM!!

LOSS = AVERAGE CROSS-ENTROPY

$$\mathcal{L} = \frac{1}{N} \sum_i D(S(WX_i + b), L_i)$$


weight 1
SMALL
LOSS ℒ
LARGE!
weight 2

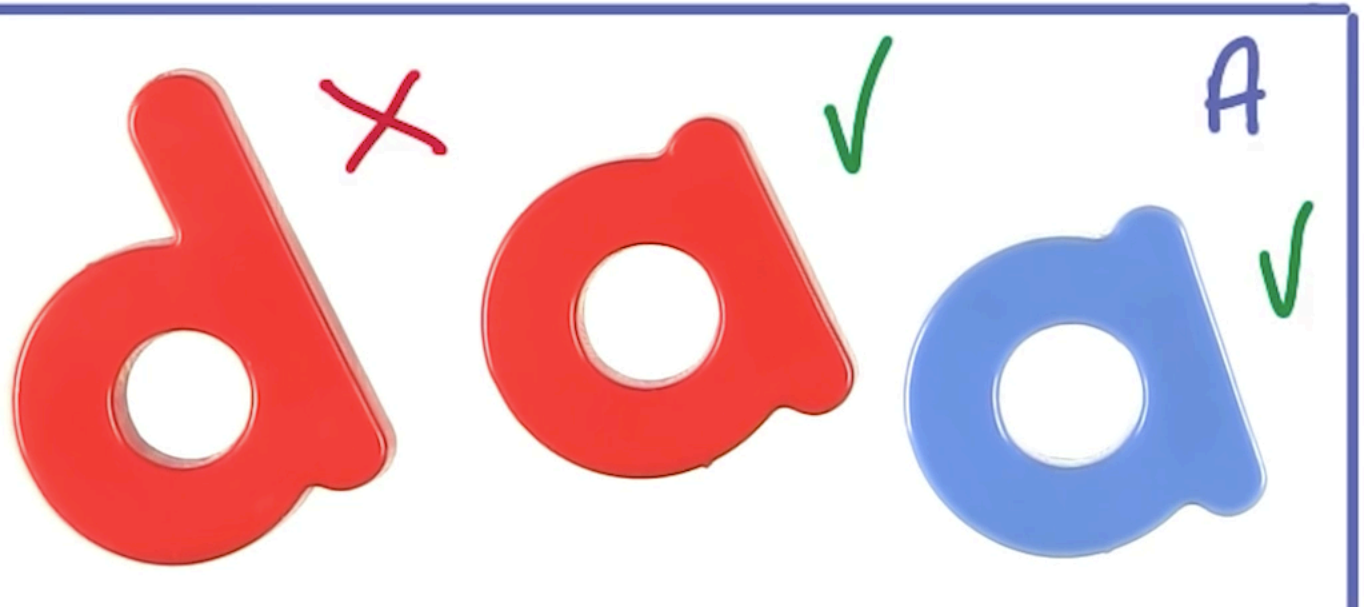

GRADIENT DESCENT
weight1
LOSS ℒ
ℒ(w1,w2)
STEP
−α ΔℒL(w1,w2)
DERIVATIVE
weight 2

TRAINING

VALIDATION

TESTING

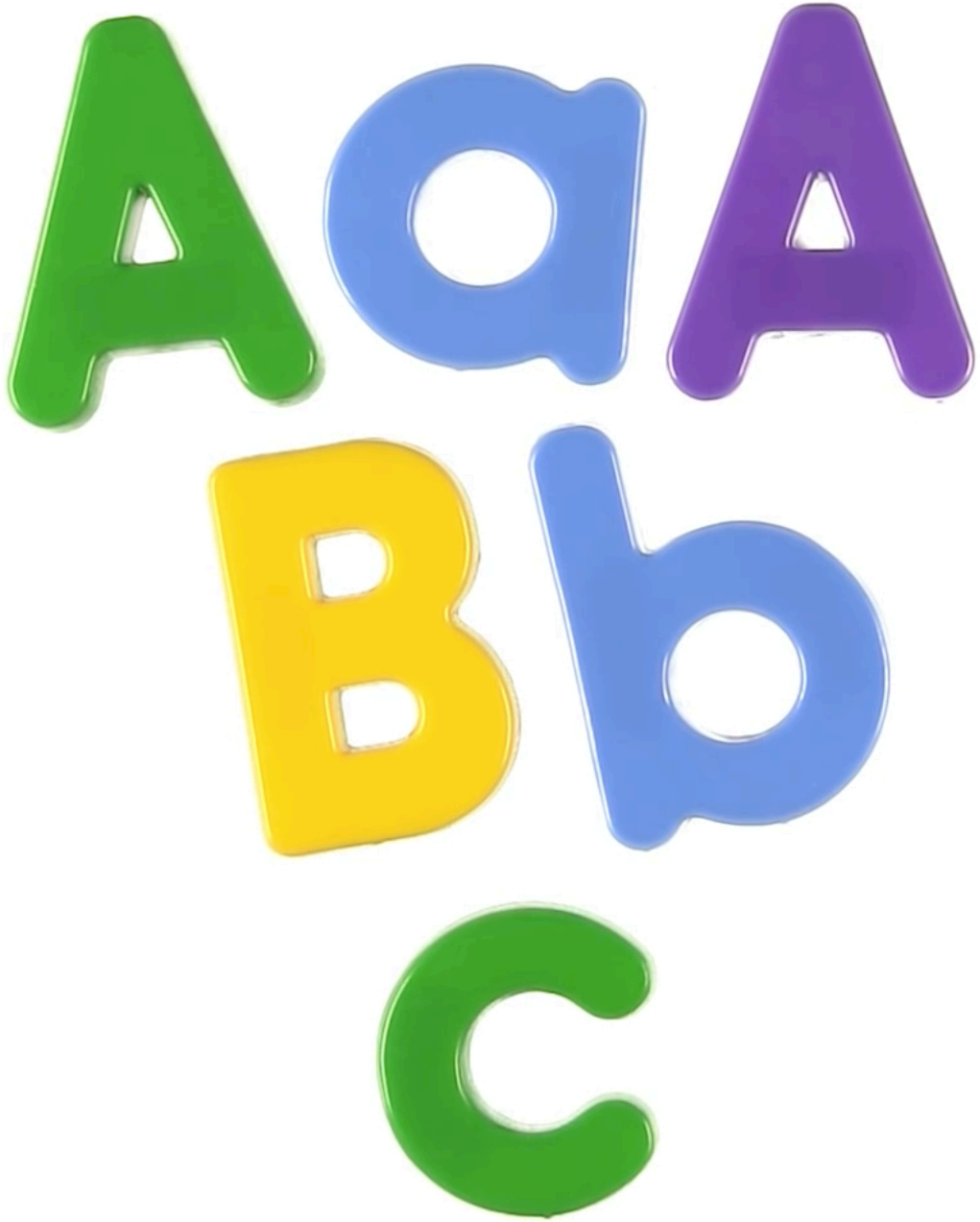

B
b
C
X
✓
✓
B


d
X
a
✓
A
a
✓

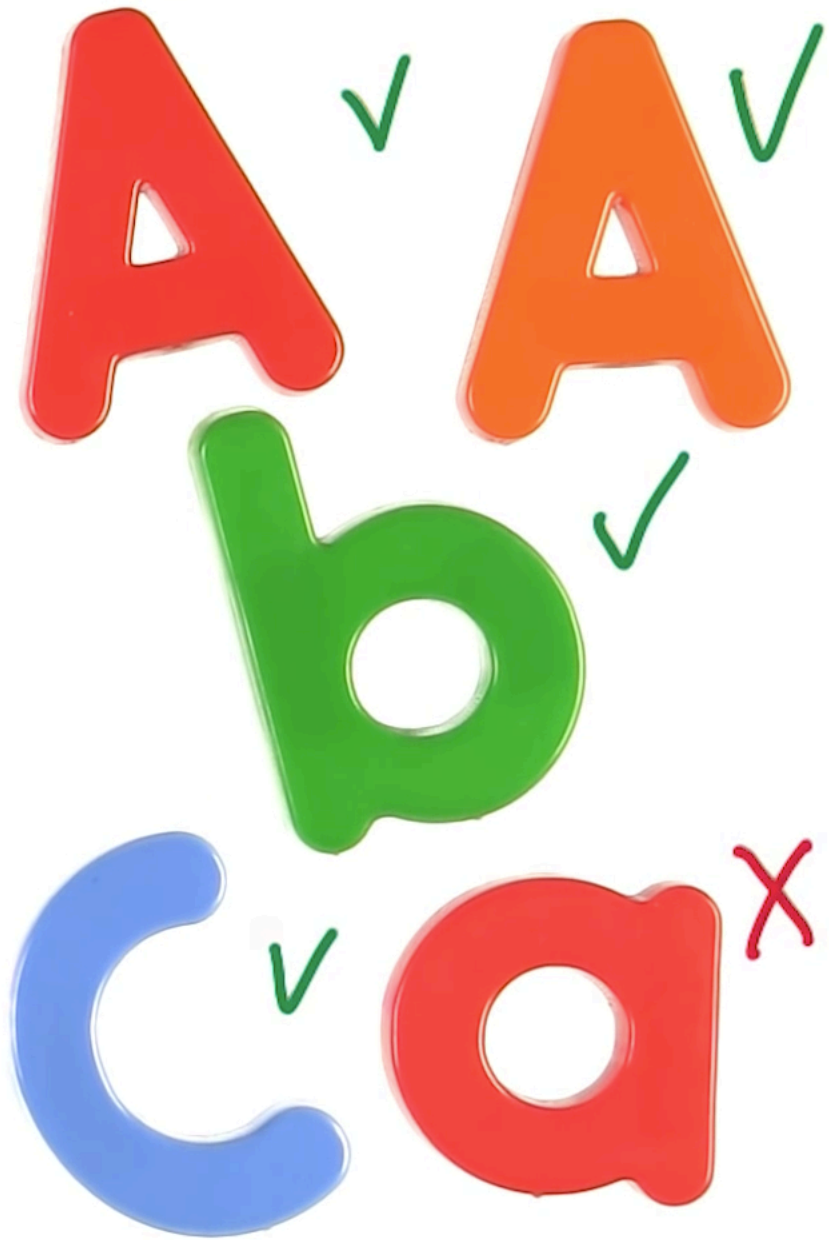

A
?


C
b
X
✓
C

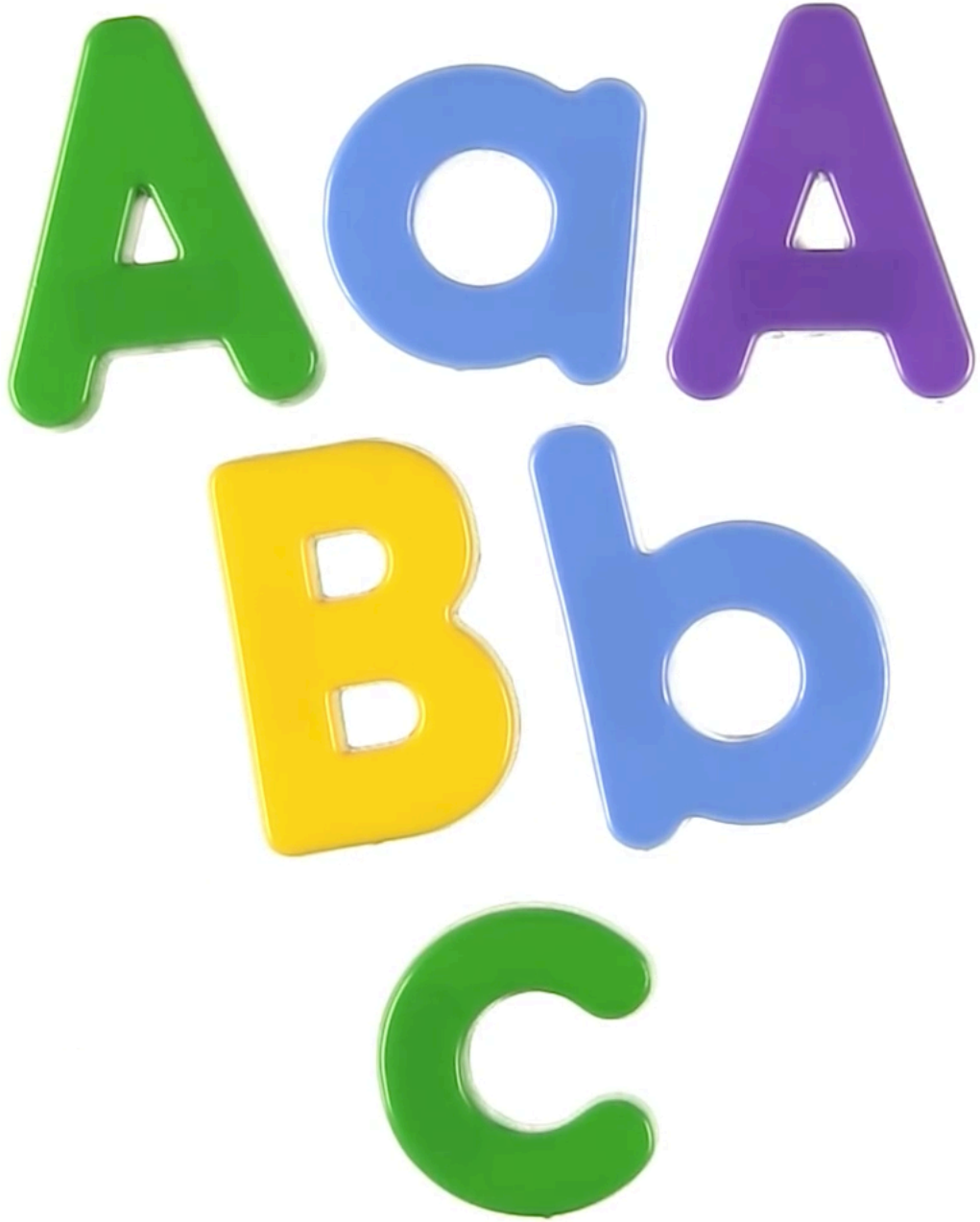
TRAINING



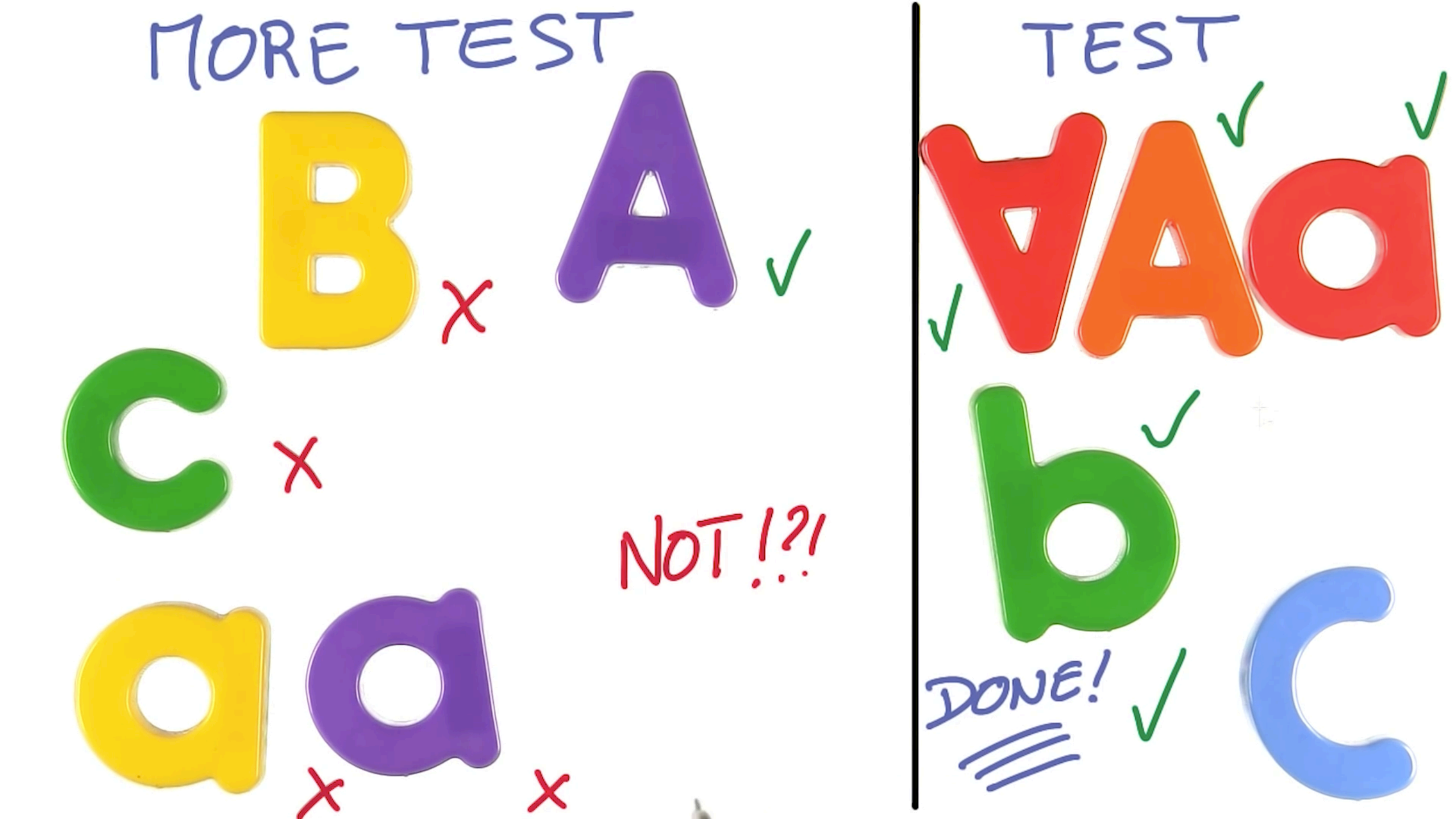
TEST

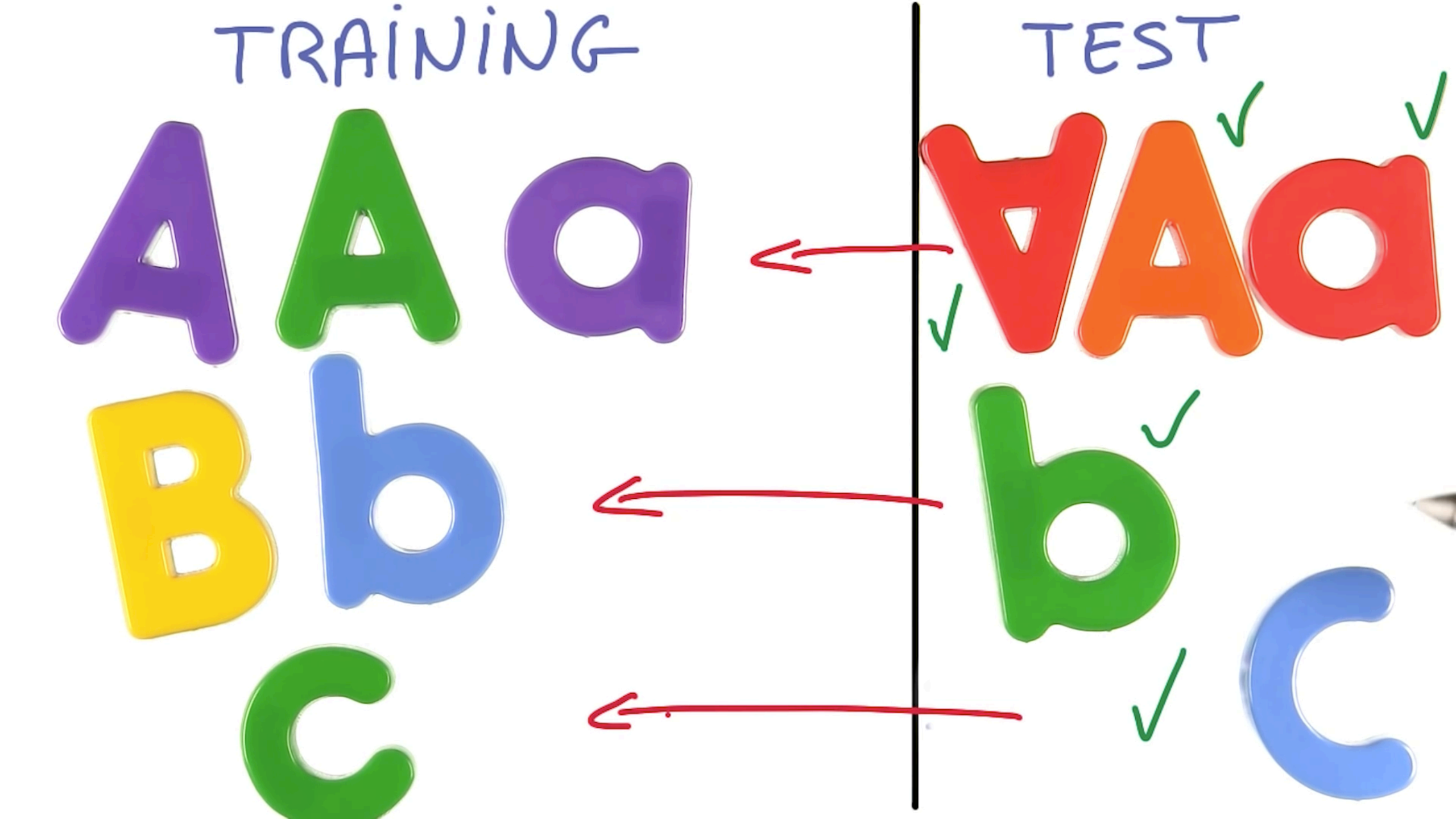


TRAINING



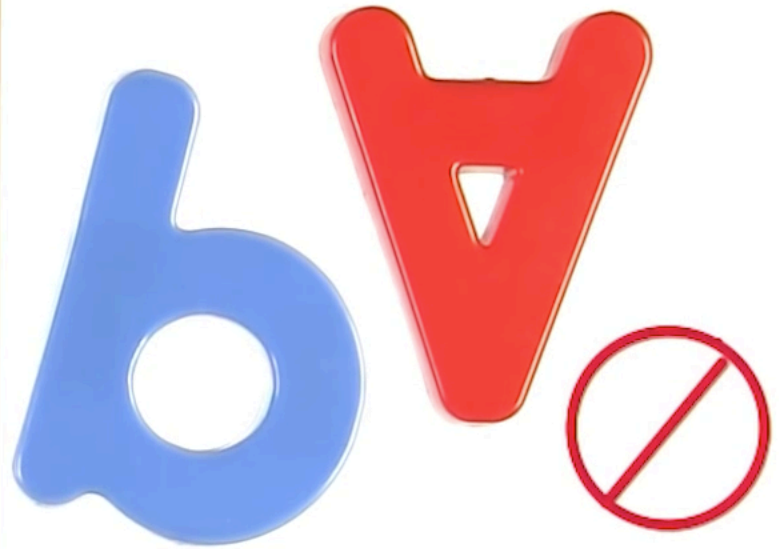
TEST

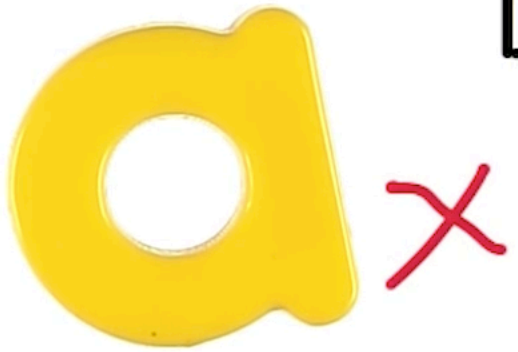

MORE TEST
B
A
C
a
a
NOT!?!
TEST
b
C
DONE!


TRAINING
TEST

TRAINING

TEST





VALIDATION

VALIDATION SET SIZE

A/

B/

✓ ✓ ✓ ✓ X X 66.%

VALIDATION SET SIZE

A/

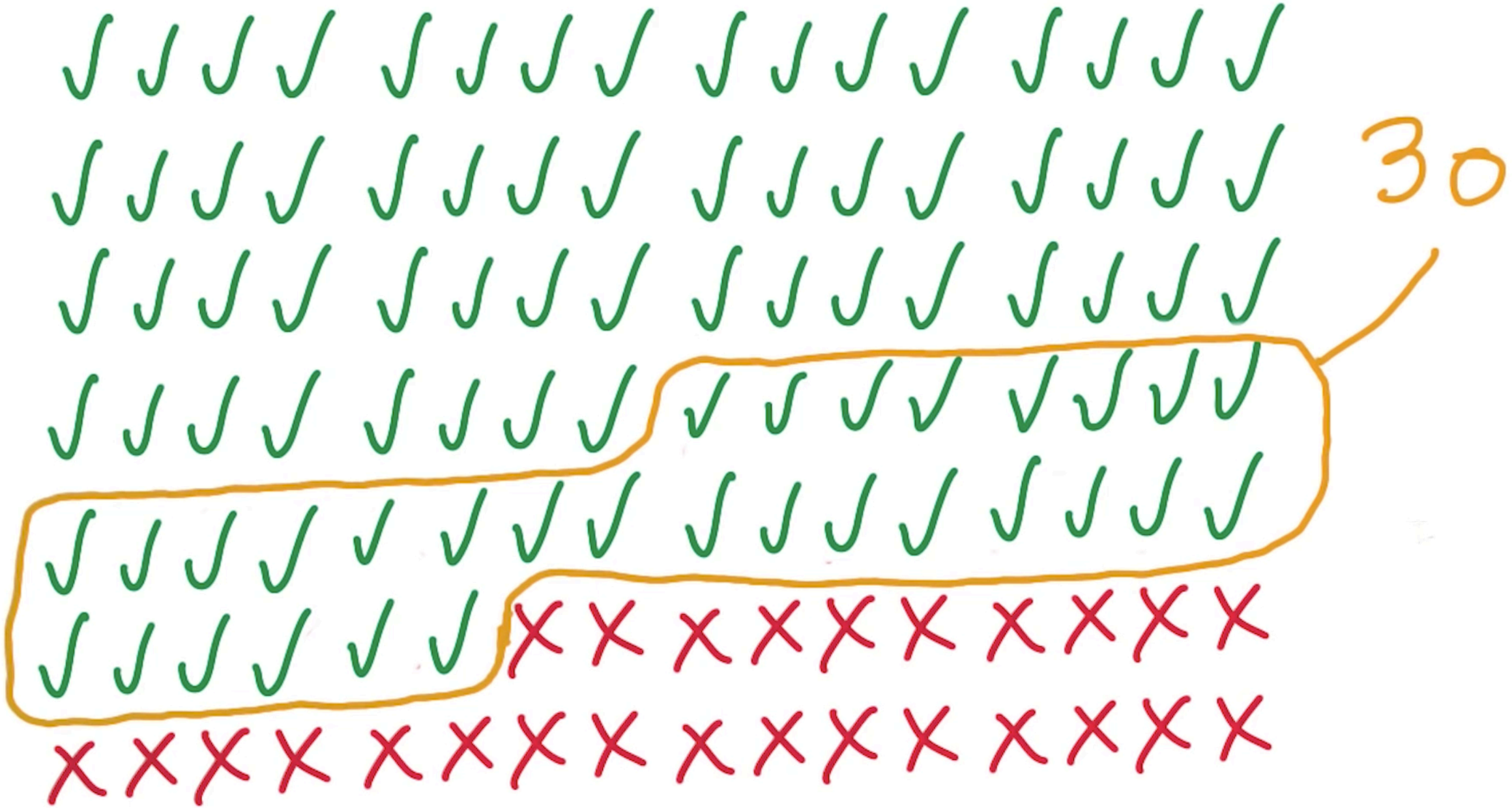
B/

✓ ✓ ✓ ✓ X X 66%

✓ ✓ ✓ ✓ ✓ X 83%


VALIDATION SET SIZE
30

VALIDATION SET SIZE



RULE OF '30'

3000 EXAMPLES

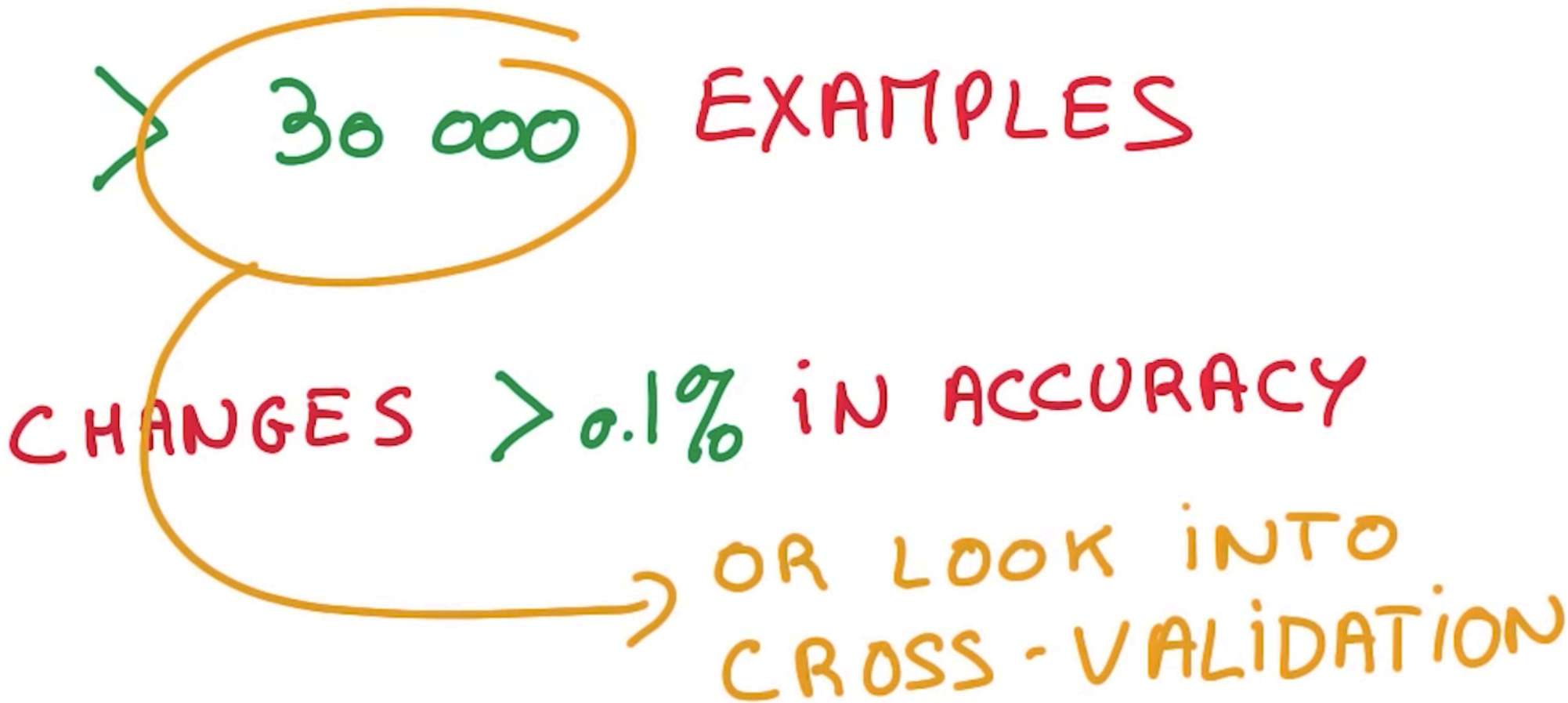
YES NO

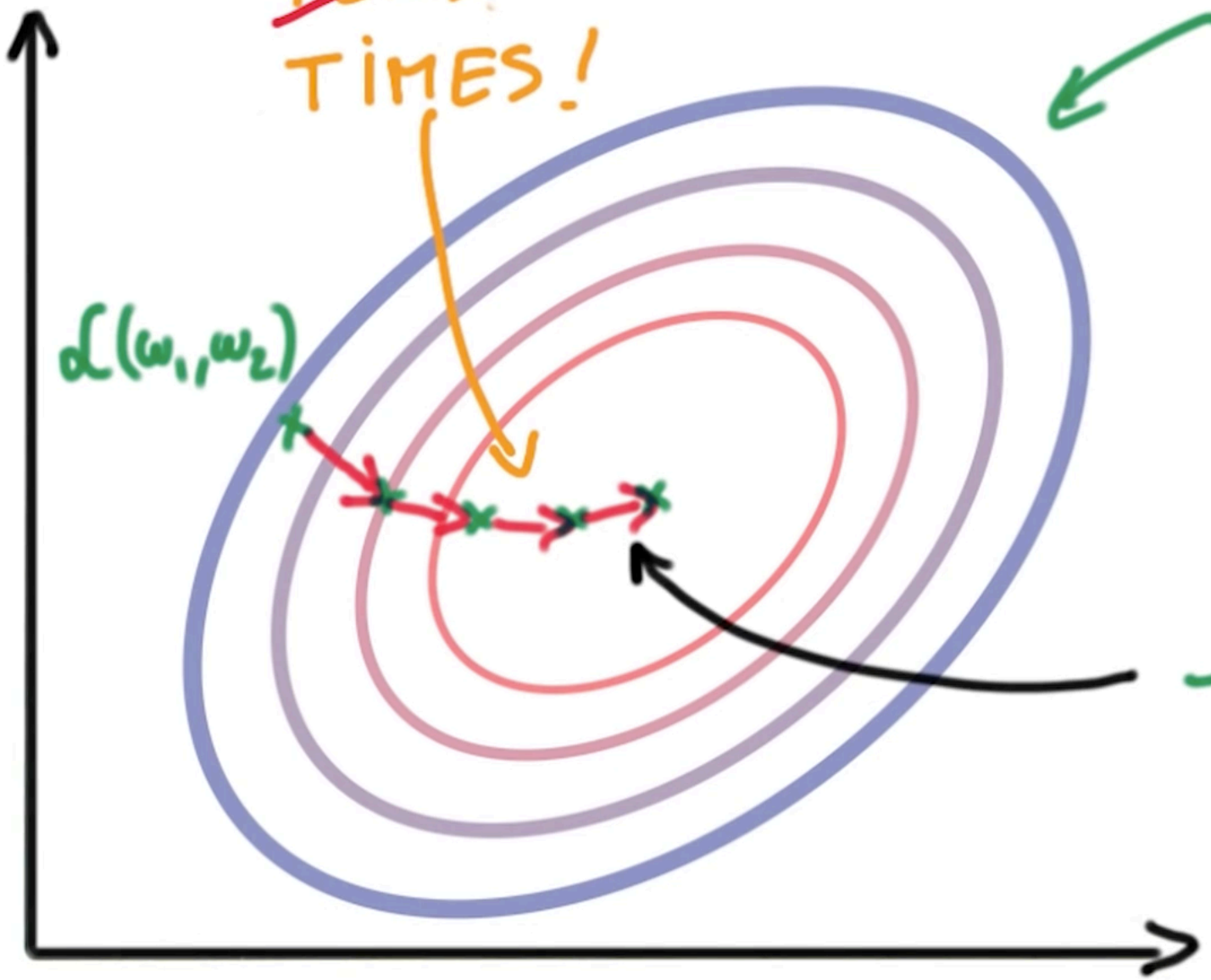
$\frac{1.0 \times 3000}{100} = 30$ ✓ ○ 80% → 81%

$\frac{0.5 \times 3000}{100} = 15$ ○ ✓ 80% → 80.5%

$\frac{0.1 \times 3000}{100} = 3$ ○ ✓ 80% → 80.1%

VALIDATION SET SIZE



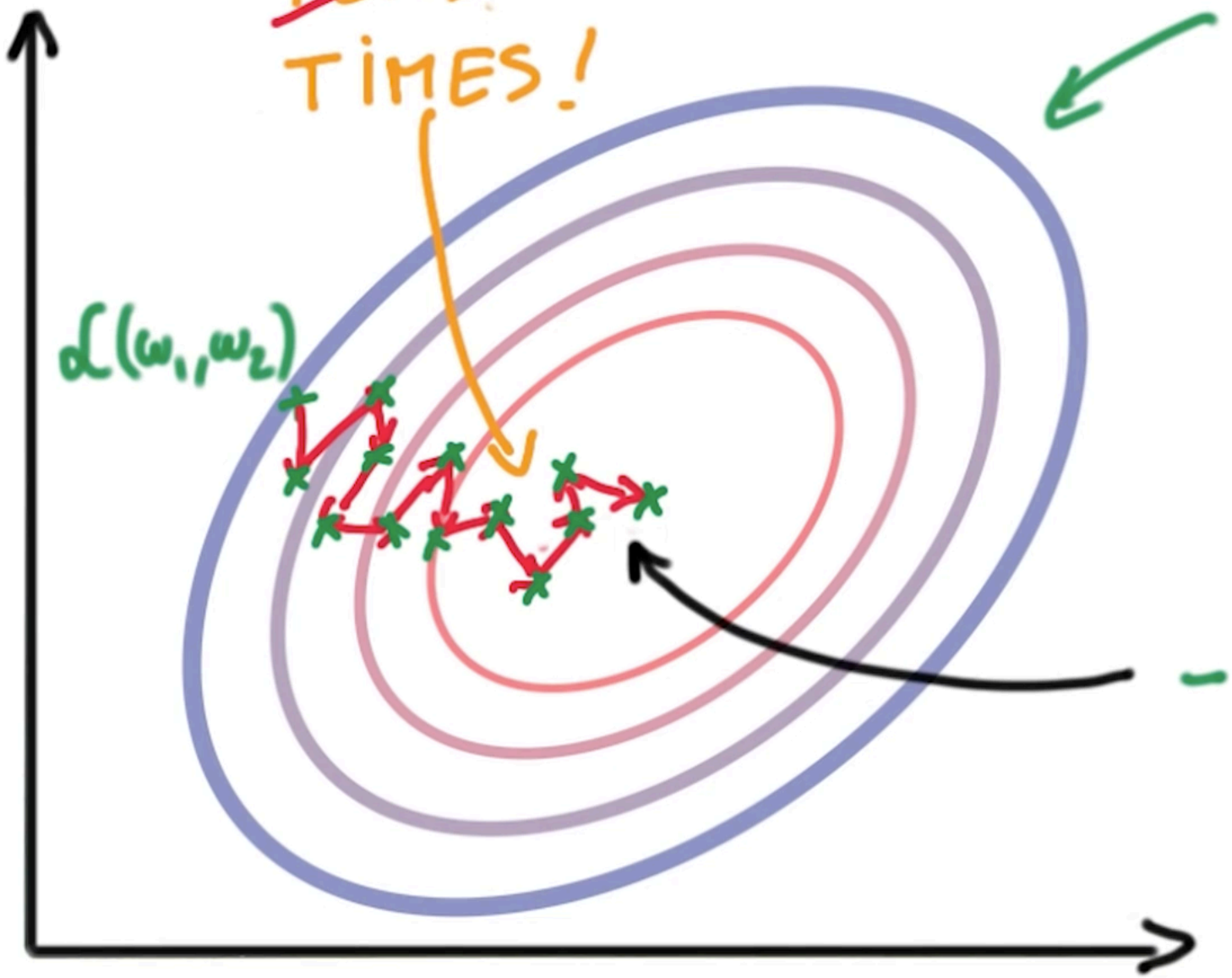


$\mathcal{L} = \sum_i D_i$

A TINY RANDOM SAMPLE OF ~~ALL~~ THE DATA!

3x THE COMPUTE

$-\alpha \Delta \mathcal{L}(w_1, w_2)$


100's TO 1000's
~~TENS~~ OF
TIMES!
$\mathcal{L}(w_1, w_2)$
$\mathcal{L} = \sum_i D_i$
A TINY
RANDOM
SAMPLE OF
~~ALL~~ THE
DATA!
3x THE COMPUTE
$-\alpha \Delta \mathcal{L}(w_1, w_2)$

STOCHASTIC GRADIENT DESCENT

S.G.D.

HELPING SGD

1- INPUTS

MEAN = ∅

EQUAL VARIANCE (SMALL)

2- INITIAL WEIGHTS

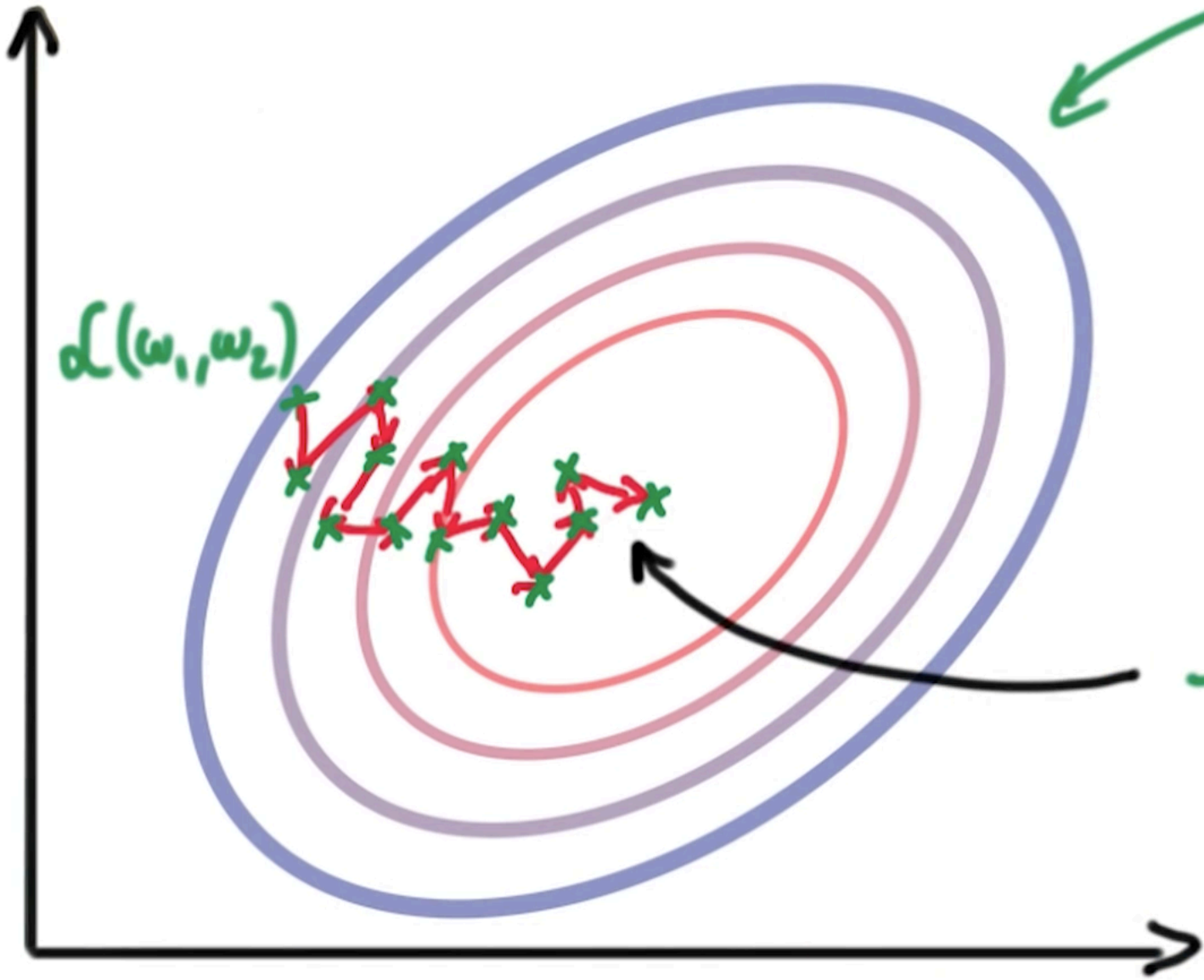
RANDOM!

MEAN = ∅

EQUAL VARIANCE (SMALL TOO)

MOMENTUM

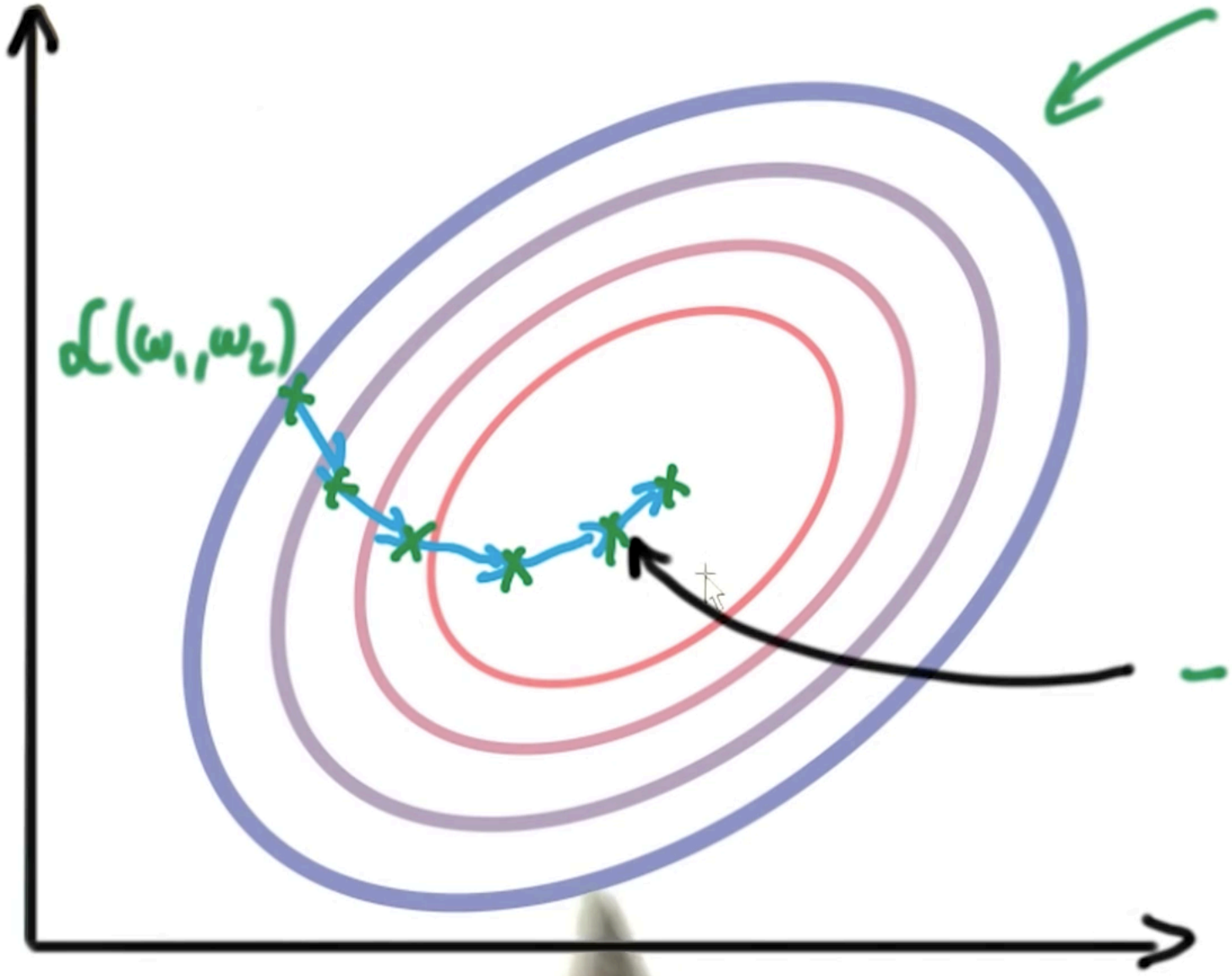
$$\mathcal{L} = \sum_i D_i$$



$$-\alpha \Delta \mathcal{L}(w_1, w_2)$$

MOMENTUM

$$\mathcal{L} = \sum_i D_i$$

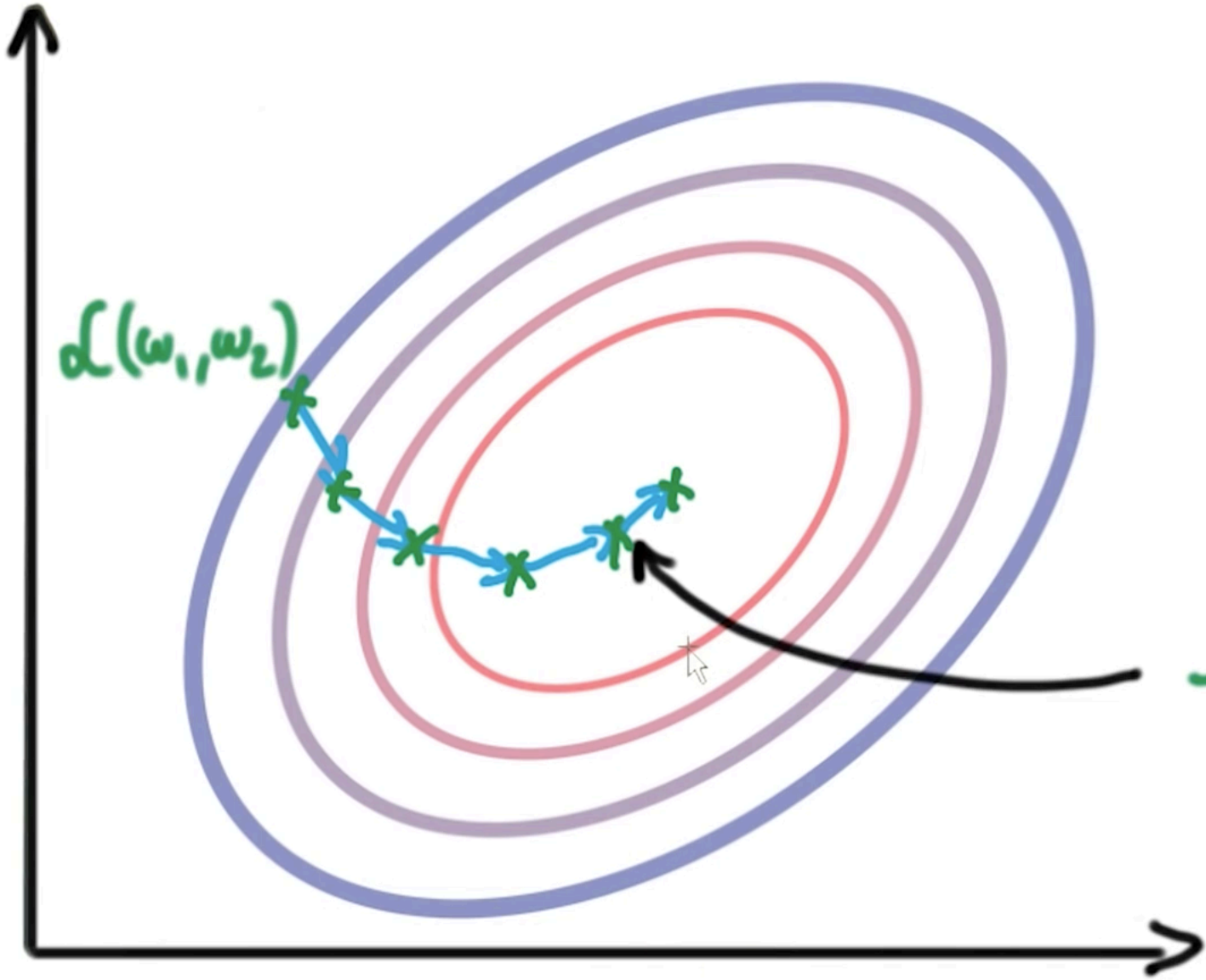


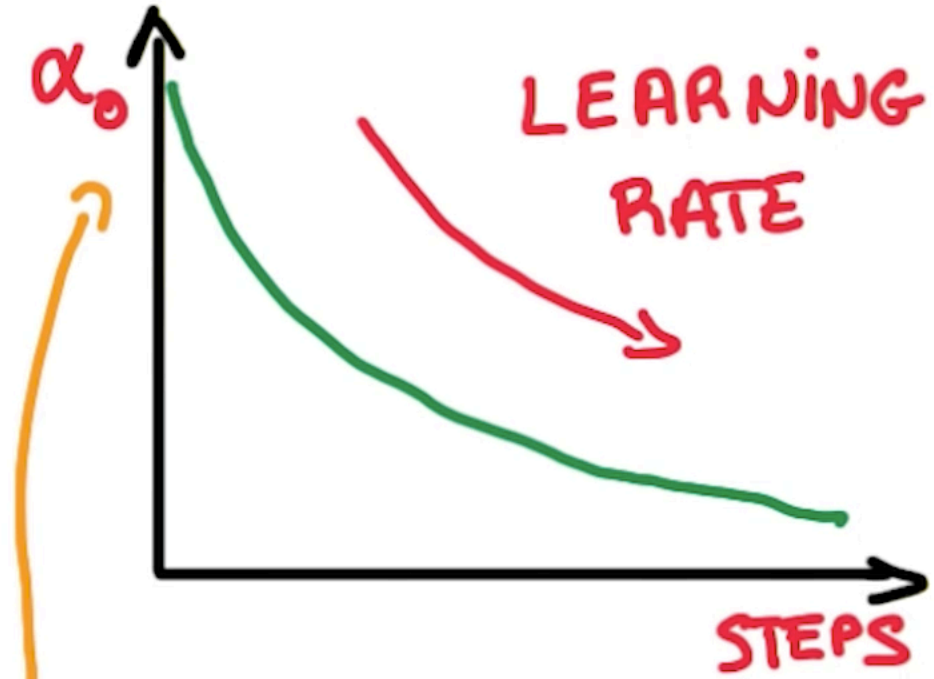
RUNNING AVERAGE

$$M \leftarrow 0.9M + \Delta\mathcal{L}$$

$$- \alpha \Delta\mathcal{L}(\omega_1, \omega_2) \; M(\omega_1, \omega_2)$$

LEARNING RATE DECAY

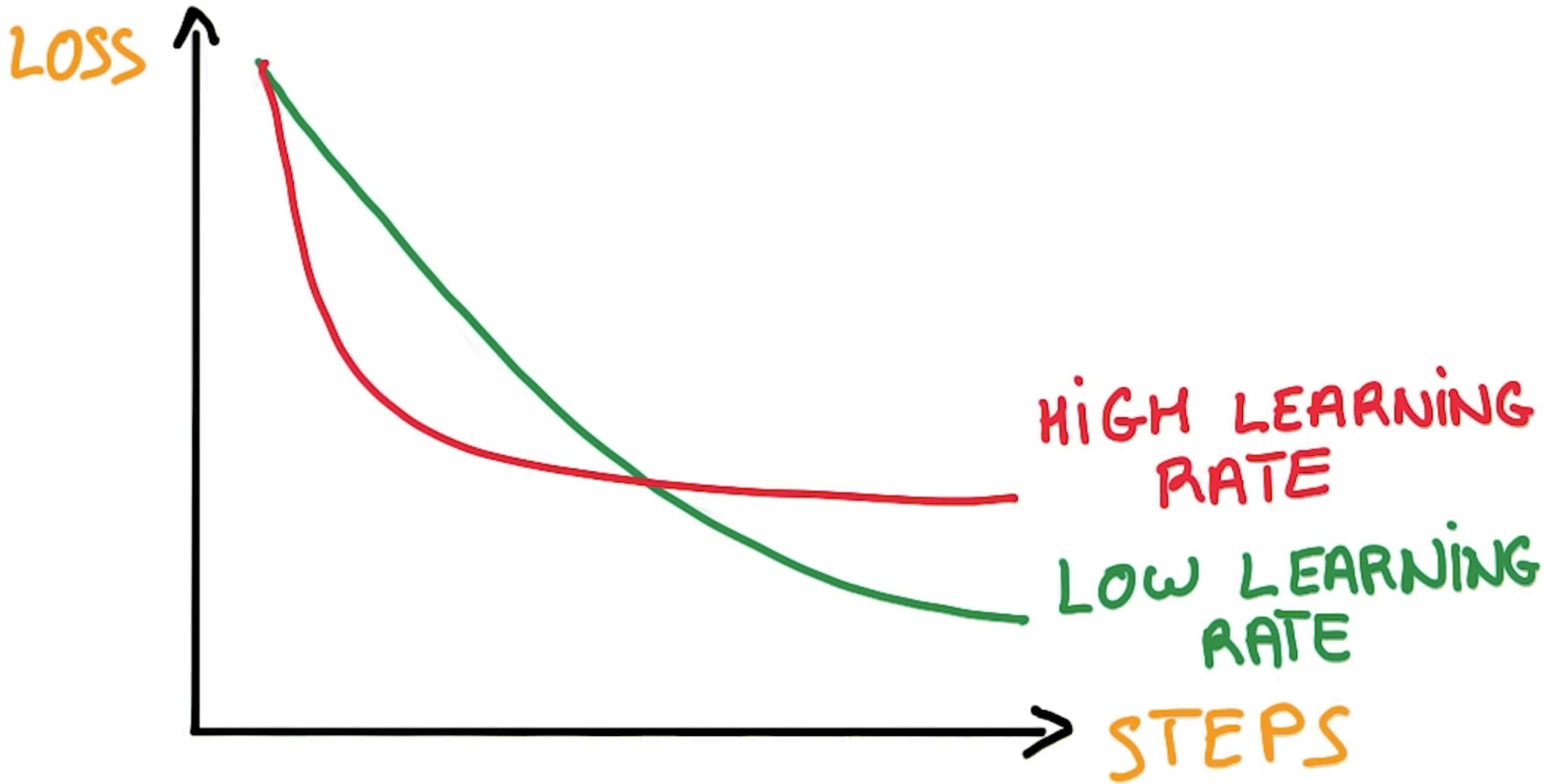




$$-\alpha \Delta(\omega_1, \omega_2)$$

↳ DERIVATIVE OR MOMENTUM

LEARNING RATE TUNING



SGD 'BLACK MAGIC'

MANY HYPER-PARAMETERS

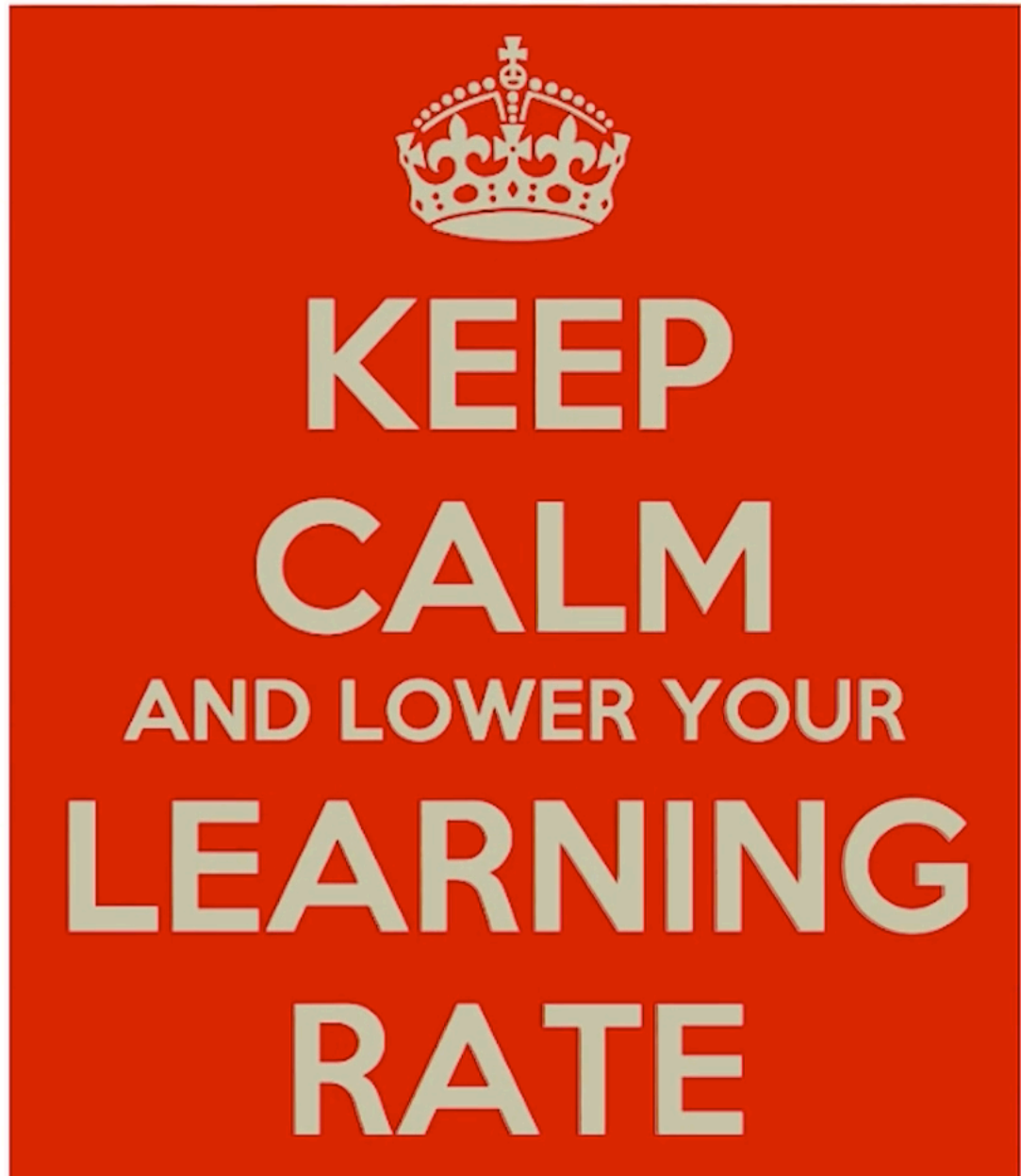
- INITIAL LEARNING RATE
- LEARNING RATE DECAY
- MOMENTUM
- BATCH SIZE
- WEIGHT INITIALIZATION

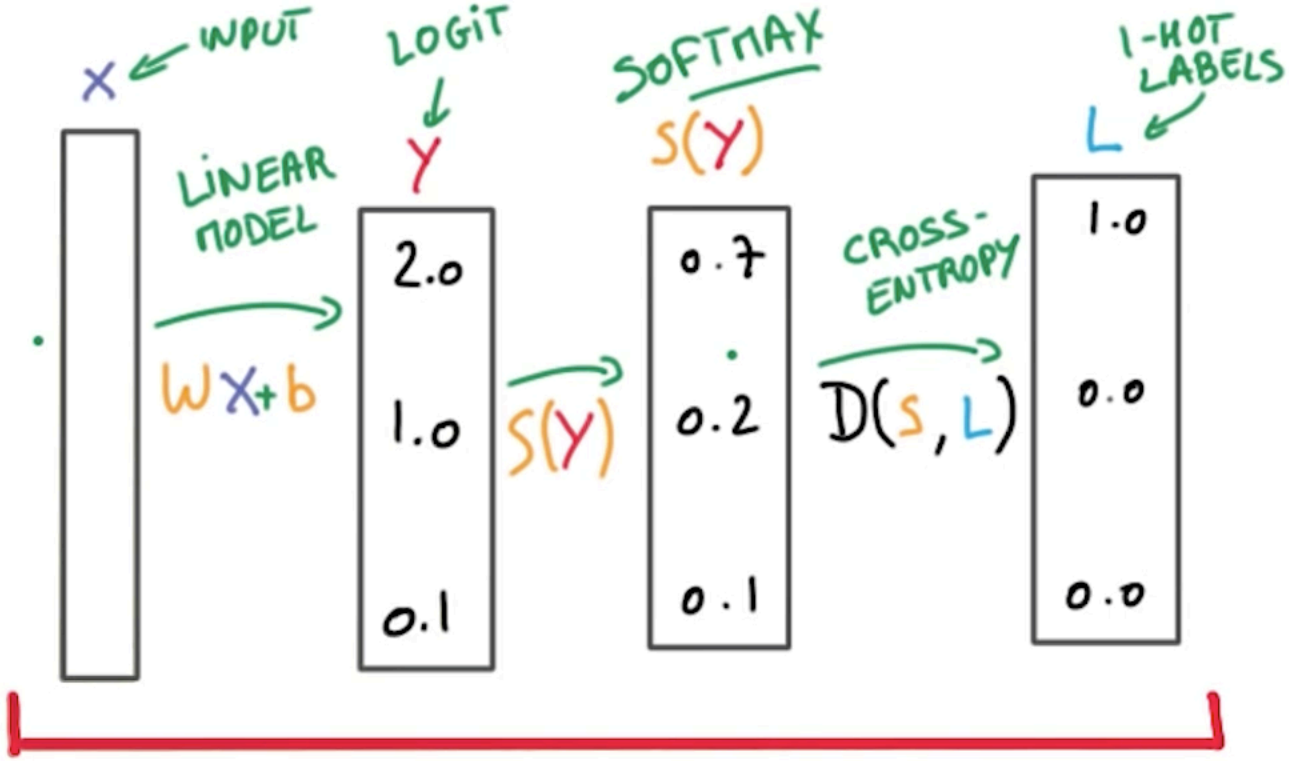
SGD 'BLACK MAGIC'

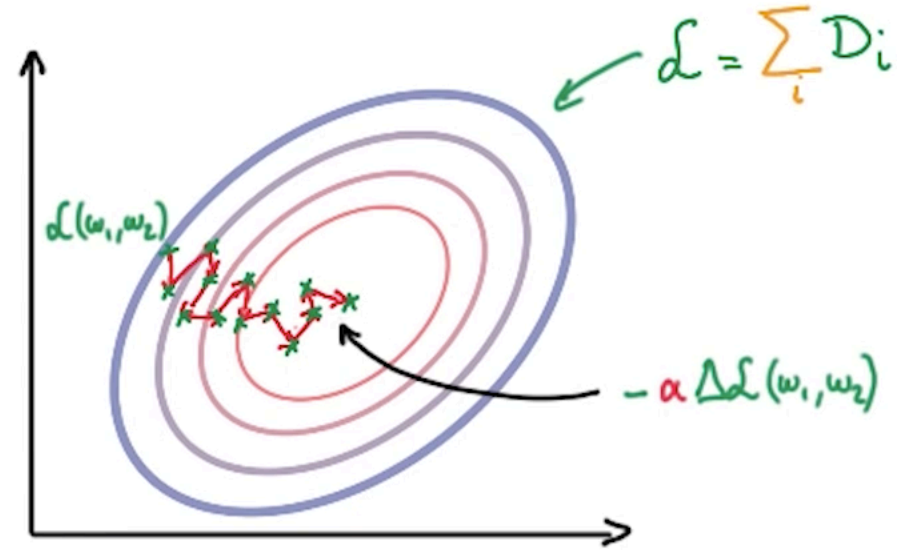
MANY HYPER-PARAMETERS

- ~~INITIAL LEARNING RATE~~
- ~~LEARNING RATE DECAY~~
- ~~MOMENTUM~~
- BATCH SIZE
- WEIGHT INITIALIZATION

ADAGRAD







'SHALLOW' MODEL

INPUT → LINEAR → OUTPUT

NO DEEP LEARNING YET!