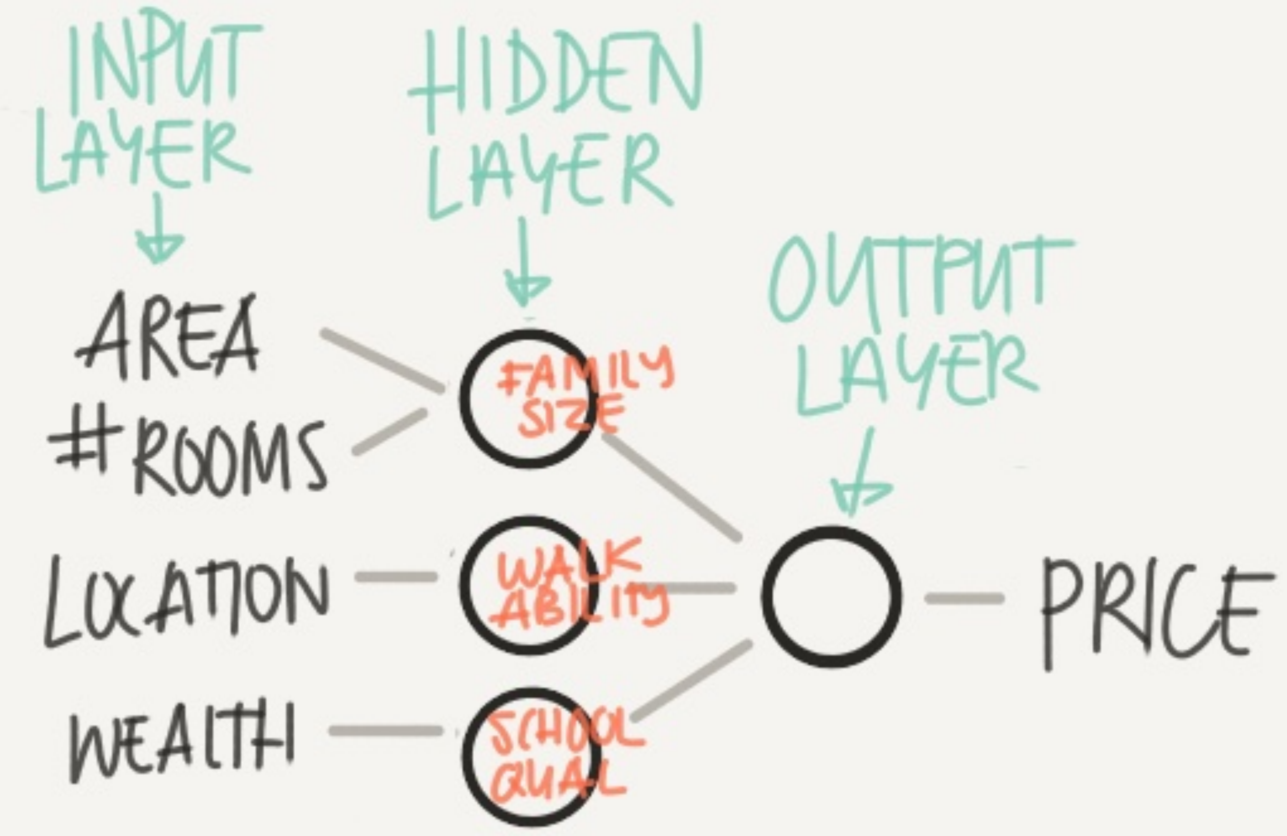
INTRO TO **DEEP LEARNING**

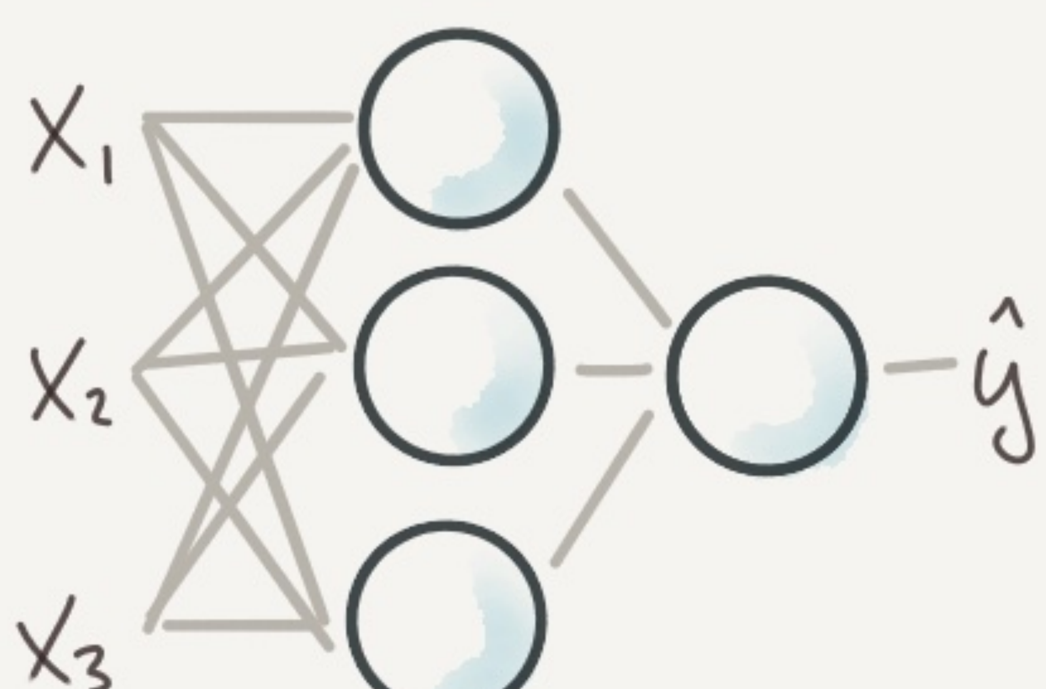
NEURAL NETWORKS & DEEP LEARNING • COURSERA • ANDREW NG • WK 1



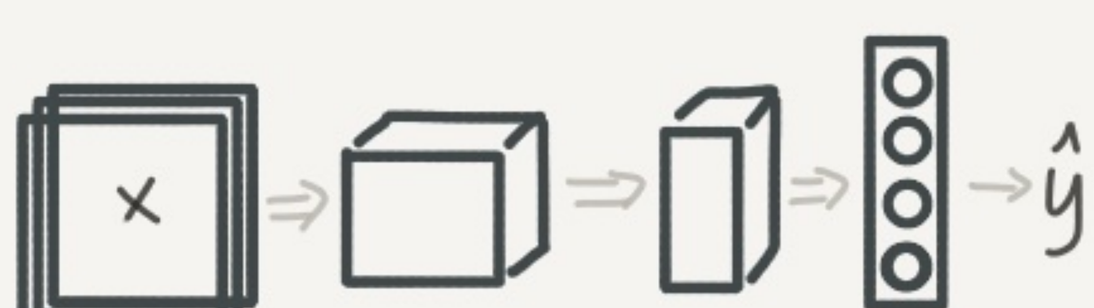
SUPERVISED LEARNING

INPUT: X	OUTPUT: y	NN TYPE
HOME FEATURES AD+USER INFO	PRICE WILL CLICK ON AD (0/1)	STANDARD NN
IMAGE	OBJECT (1...1000)	CONV. NN (CNN)
AUDIO ENGLISH	TEXT TRANSCRIPT CHINESE	RECURRENT NN (RNN)
IMAGE/RADAR	POS OF OTHER CARS	CUSTOM/HYBRID

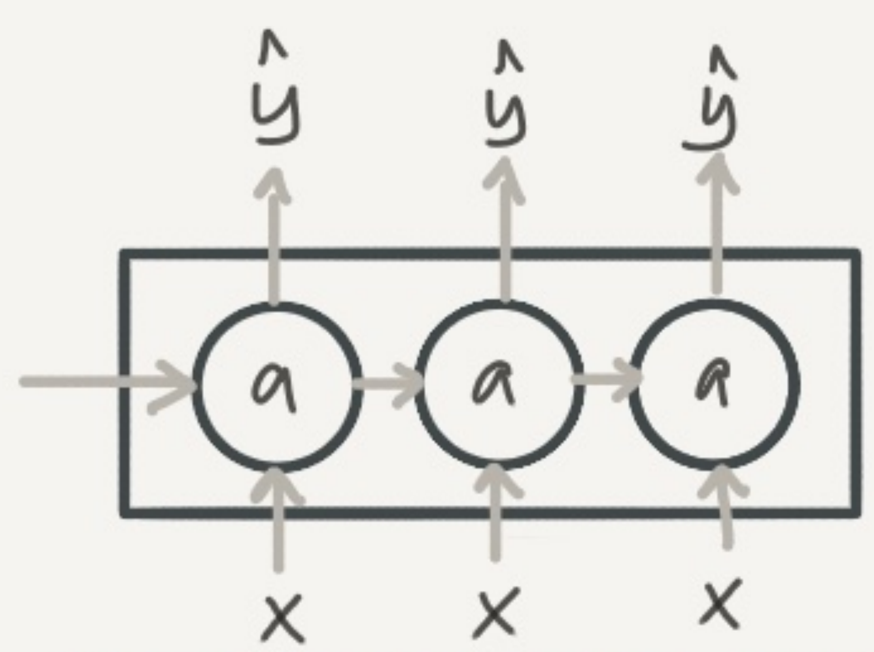
NETWORK ARCHITECTURES



STANDARD NN



CONVOLUTIONAL NN



RECURRENT NN

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA

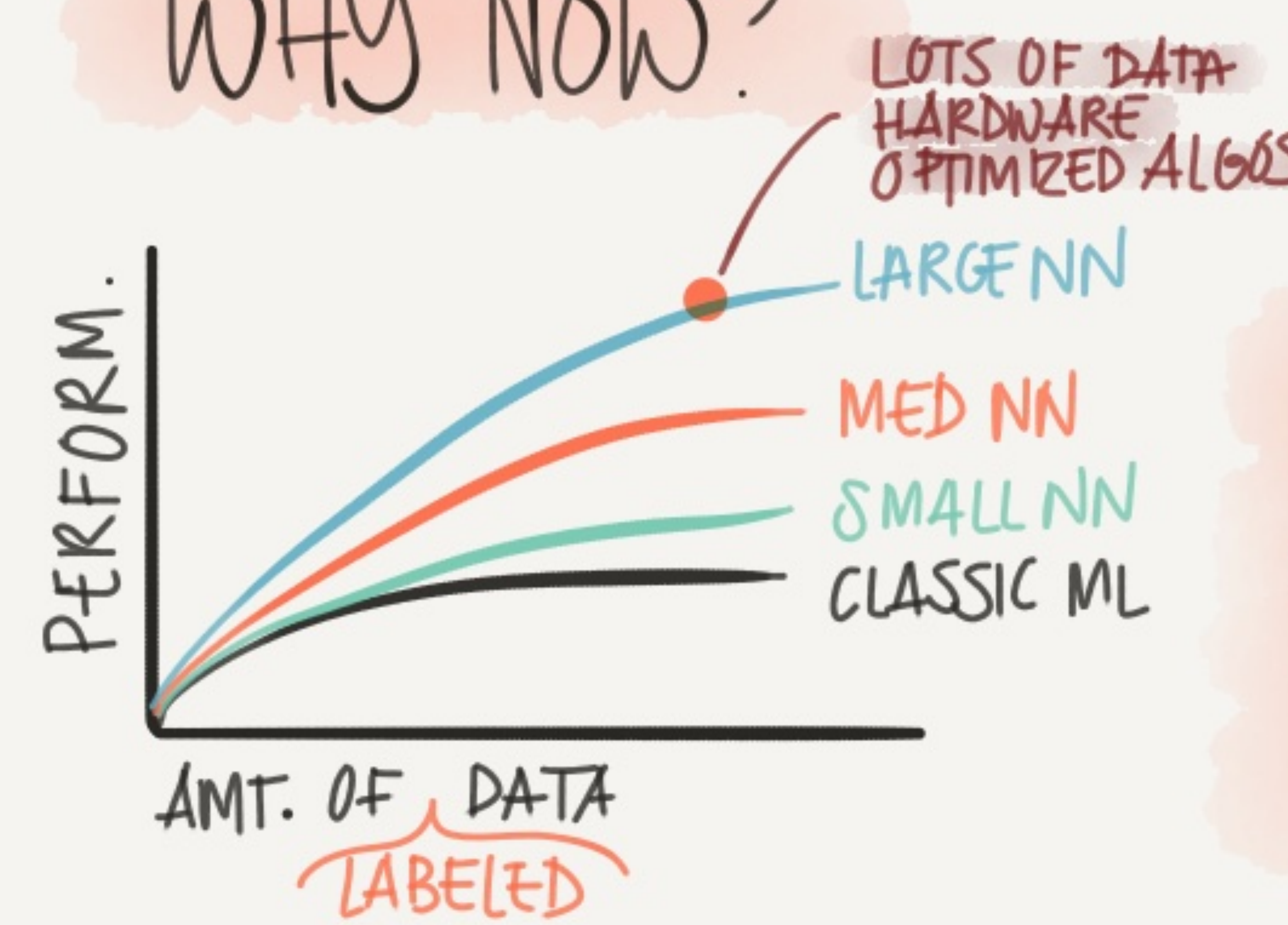
STRUCTURED

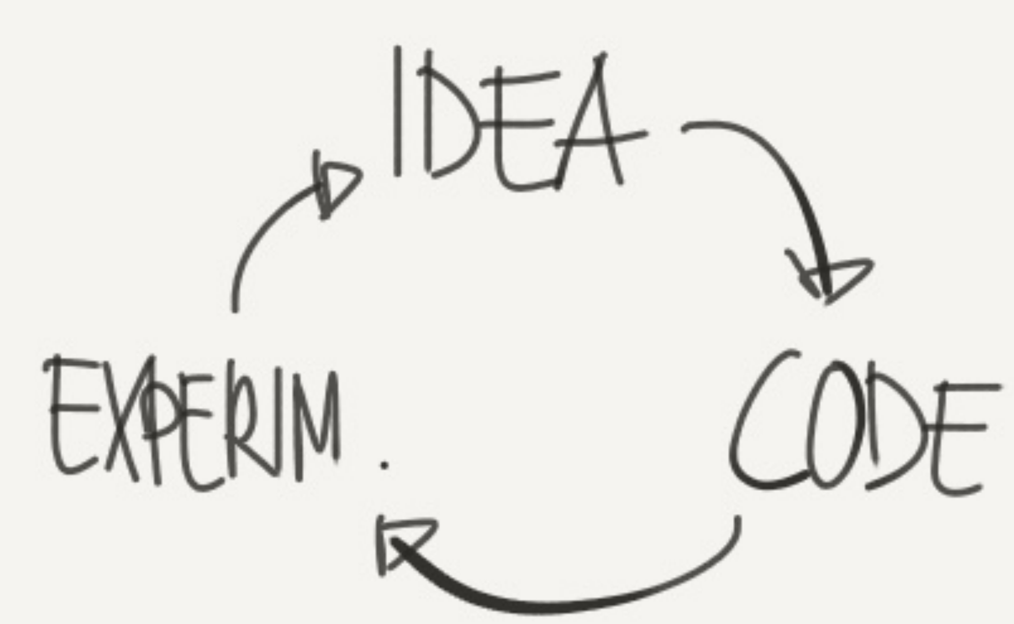
"THE QUICK BROWN FOX"

UNSTRUCTURED

HUMANS ARE GOOD AT THIS

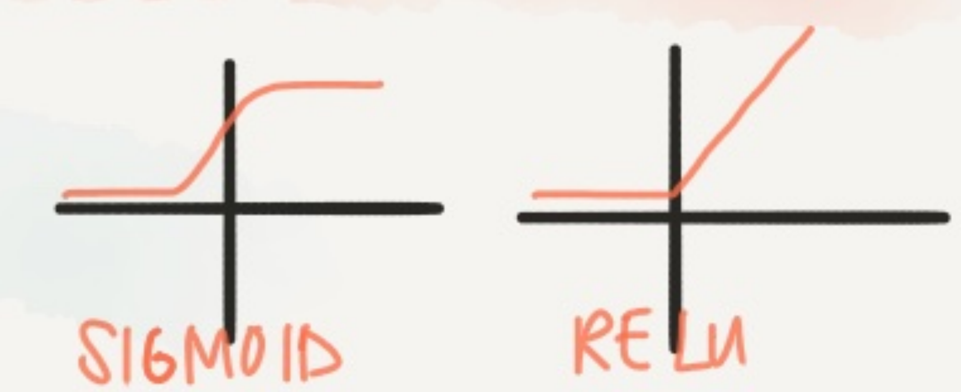
WHY NOW?





FASTER COMPUTATION IS IMPORTANT TO SPEED UP THE ITERATIVE PROCESS

ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT



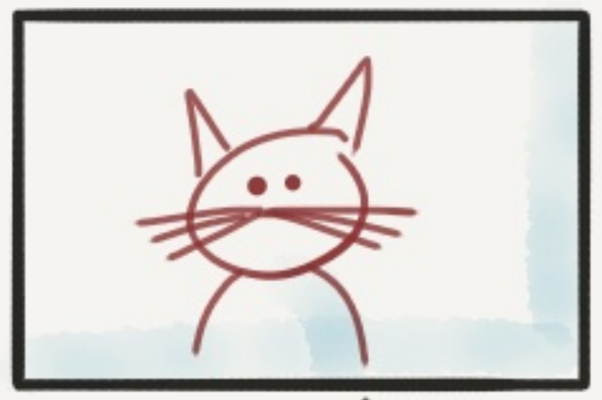
@TessFerrandez

LOGISTIC REGRESSION

AS A NEURAL NET

NEURAL NETWORKS & DEEP LEARNING • COURSERA • ANDREW NG • WK 2

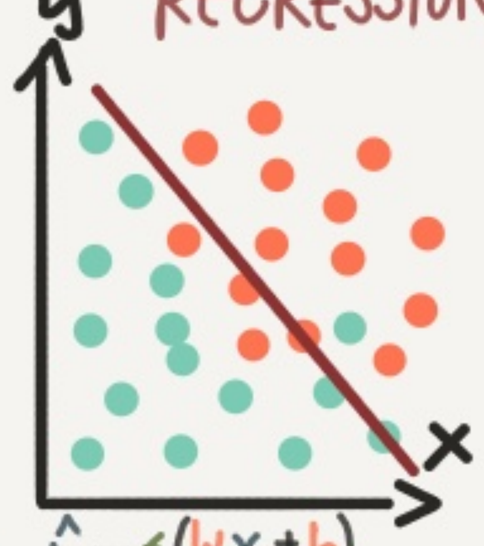
BINARY CLASSIFICATION



→ 1 : CAT
0 : NOT CAT

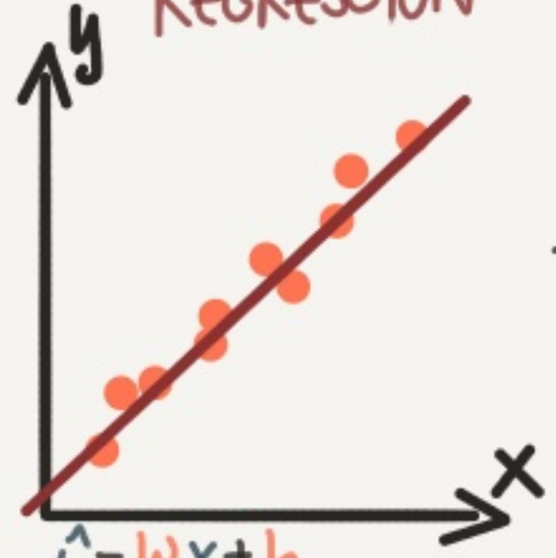
X y

LOGISTIC REGRESSION = LINEAR REGRESSION + σ SIGMOID



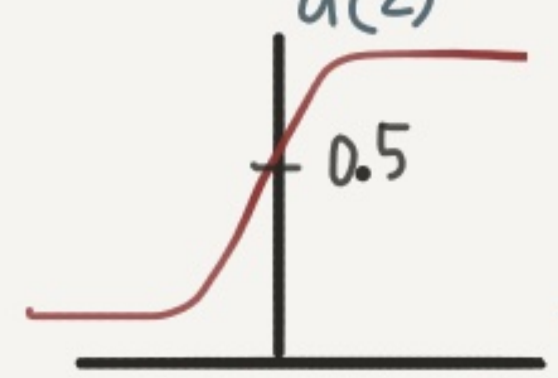
$\hat{y} = \sigma(wx + b)$

=



$\hat{y} = wx + b$

+



$a(z) = 1$ FOR LARGE Z
0 FOR SMALL Z

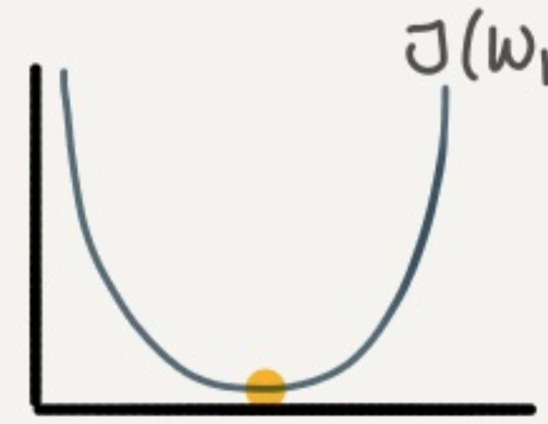
THE TASK IS TO LEARN w & b BUT HOW?

A : OPTIMIZE HOW GOOD THE GUESS IS BY MINIMIZING THE DIFF BETWEEN GUESS ($\hat{y}$) AND TRUTH (y)

$\text{LOSS} = \mathcal{L}(\hat{y}, y)$

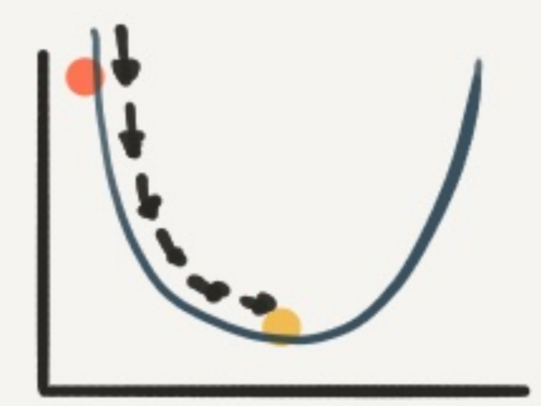
$\text{COST} = J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

COST = LOSS FOR THE ENTIRE DATASET



FIND THE MINIMUM

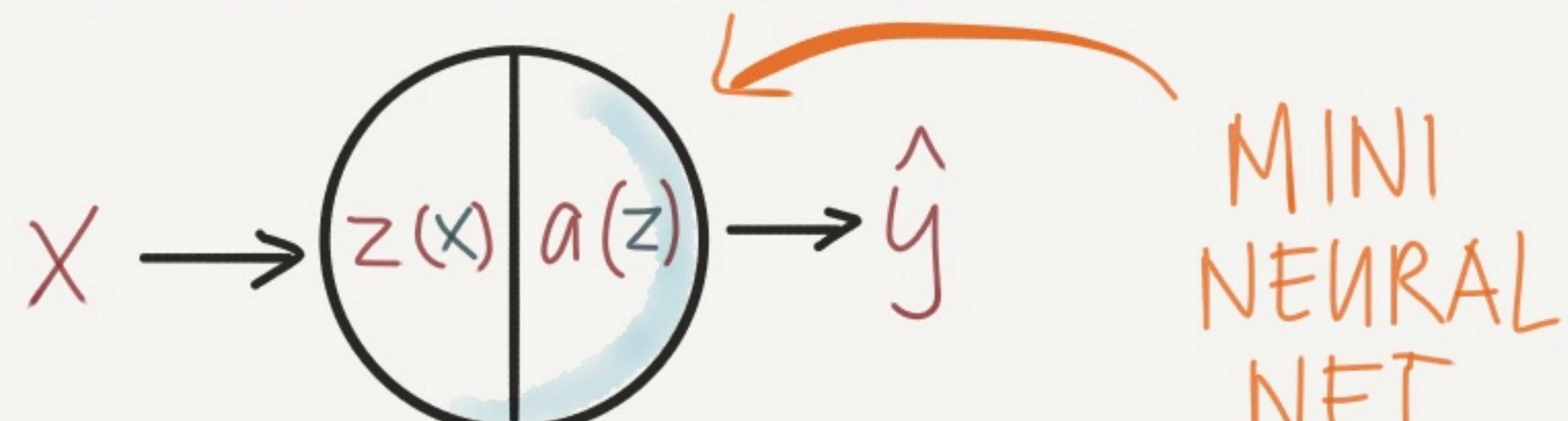
FINDING THE MINIMUM WITH GRADIENT DESCENT



1. FIND THE DOWNHILL DIRECTION (USING DERIVATIVES)
2. WALK (UPDATE W & B) AT A α LEARNING RATE

REPEAT UNTIL YOU REACH BOTTOM (CONVERGE)

PUTTING IT ALL TOGETHER



$z(x) = wx + b$

$\hat{y} = a(z) = \sigma\ \text{SIGMOID}(z)$

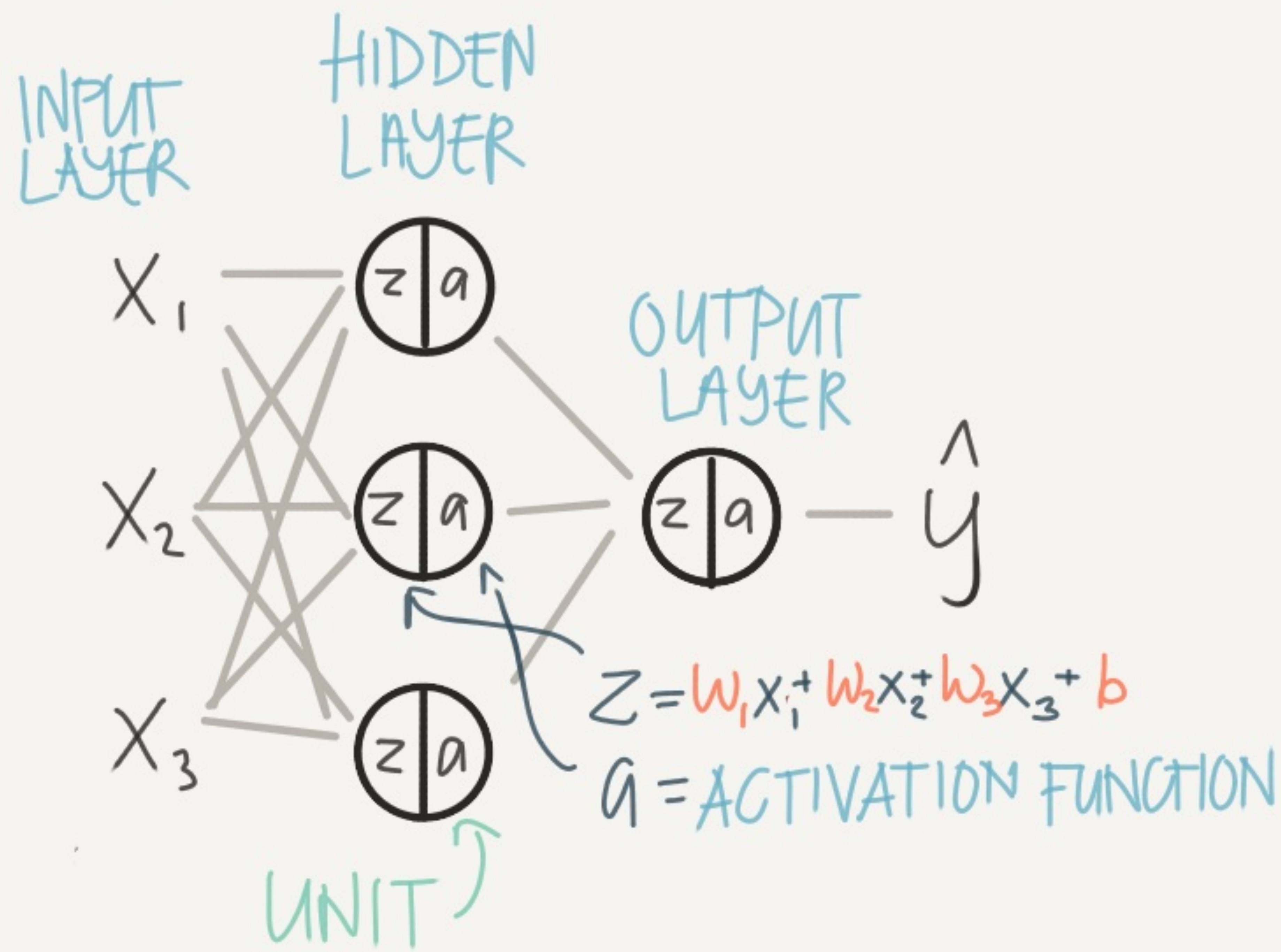
1. FORWARD PROPAGATION • CALCULATE $\hat{y}$
2. BACKWARD PROPAGATION • GRADIENT DESCENT + UPDATE w & b

REPEAT UNTIL IT CONVERGES

@TessFerrandez

SHALLOW NEURAL NETS

2 LAYER NEURAL NET



ACTIVATION FUNCTIONS

SIGMOID σ
BINARY CLASSIFIER - ONLY USED FOR OUTPUT LAYER

SLOW GRAD DESCENT SINCE SLOPE IS SMALL FOR LARGE/SMALL VAL

TANH
NORMALIZED ⇒ GRADIENT DESCENT IS FASTER

RELU
DEFAULT CHOICE FOR ACTIVATION

SLOPE = 1/0

LEAKY RELU
AVOIDS UNDEF SLOPE AT Ø BUT RARELY USED IN PRACTICE

WHY ACTIVATION FUNCTIONS?

EX. WITH NO ACTIVATION – $a = z$

$$a^{[1]} = z^{[1]} = w^{[1]}x + b^{[1]} \quad \text{LAYER 1}$$
$$a^{[2]} = z^{[2]} = w^{[2]}a^{[1]} + b^{[2]} \quad \text{LAYER 2}$$

PLUG IN $a^{[1]}$

$$a^{[2]} = w^{[2]}(w^{[1]}x + b^{[1]}) + b^{[2]}$$
$$= \underbrace{w^{[2]}w^{[1]}}_{w'}x + \underbrace{w^{[2]}b^{[1]} + b^{[2]}}_{b'}$$
$$w'x + b'$$

← LINEAR FUNCTION

WE COULD JUST AS WELL HAVE SKIPPED THE WHOLE NEURAL NET & USED LIN. REGR.

INITIALIZING W+b

WHAT IF: INIT TO Ø

THIS WILL CAUSE ALL THE UNITS TO BE THE SAME AND LEARN EXACTLY THE SAME FEATURES

SOLUTION: RANDOM INIT

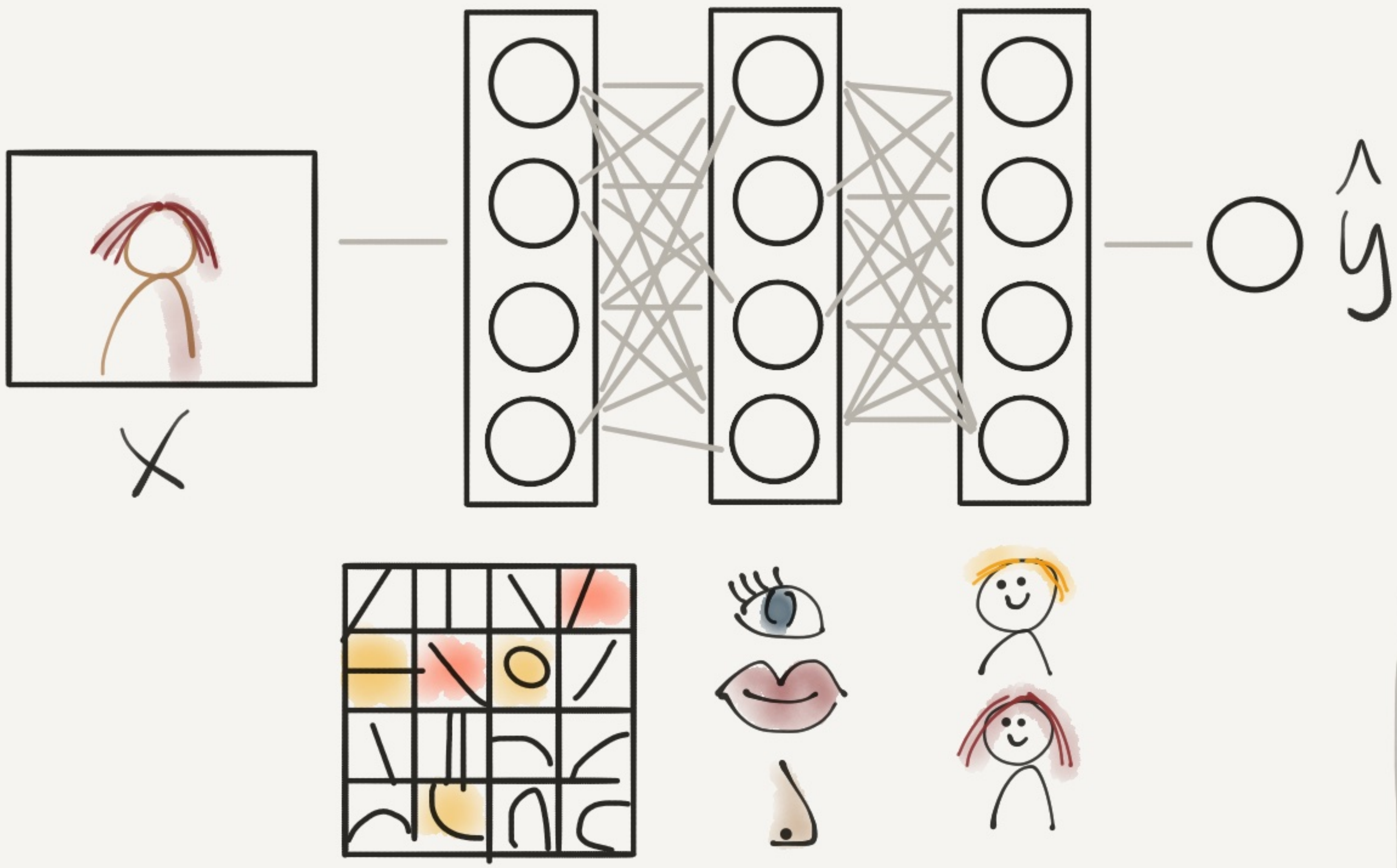
BUT ALSO WANT THEM SMALL SO RAND * 0.01 ← HYPERPARAM

@TessFerrandez

DEEP
NEURAL NETS

THERE ARE FUNCTIONS A SMALL DEEP NET CAN COMPUTE THAT SHALLOW NETS NEED EXP. MORE UNITS TO COMP.

WHY DEEP NEURAL NETS?



VERY DATA HUNGRY

NEED LOTS OF COMPUTER POWER

ALWAYS VECTORIZE

VECTOR MULT. CHEAPER THAN FOR LOOPS

COMPUTE ON GPUs

LOTS OF HYPERPARAMS

- LEARNING RATE α
- # ITERATIONS
- # HIDDEN LAYERS
- # HIDDEN UNITS
- CHOICE OF ACTIVATION
- MOMENTUM
- MINI-BATCH SIZE
- REGULARIZATION

LOW LEVEL AUDIO WAVE FEATURES — PHONEMES — WORDS — SENTENCES

↗ ↘ PITCH

C A T

NEURAL NETWORKS & DEEP LEARNING • COURSERA • ANDREW NG • WK 4