



**KEEP
CALM
AND
STUDY**

ARTIFICIAL NEURAL NETWORK

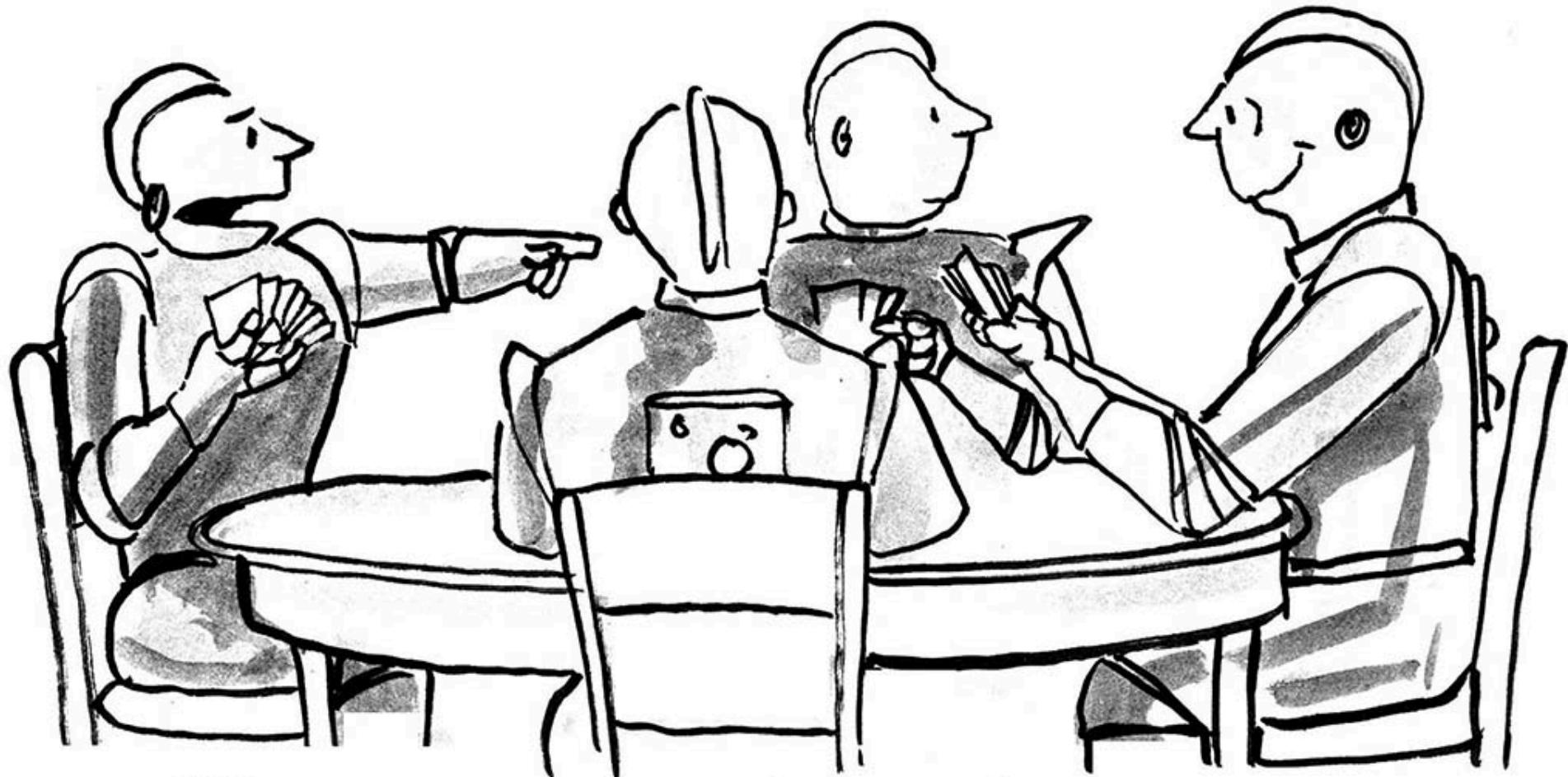


Artificial Neural Network basics

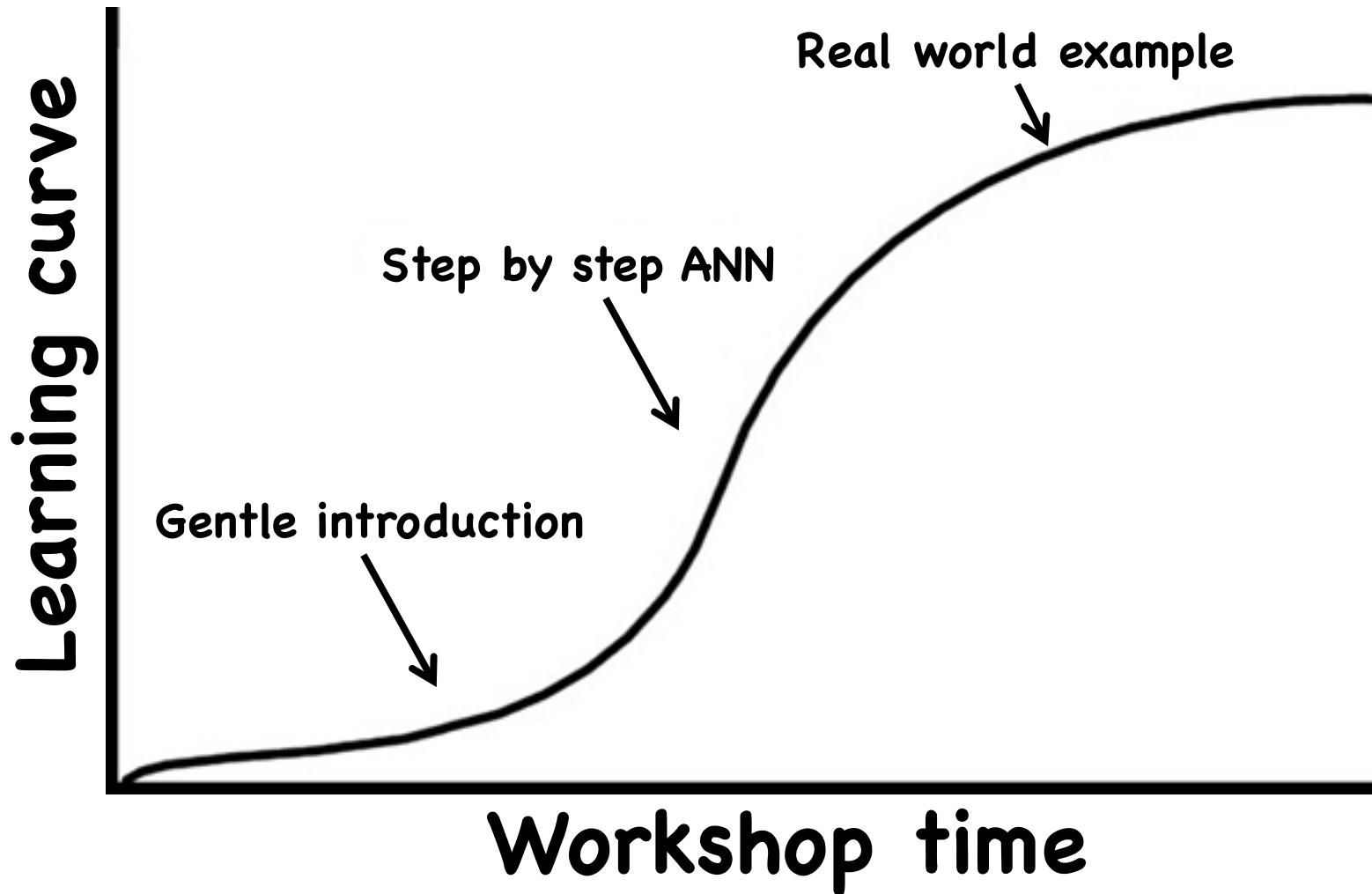
Qingkai Kong
2016-12-02



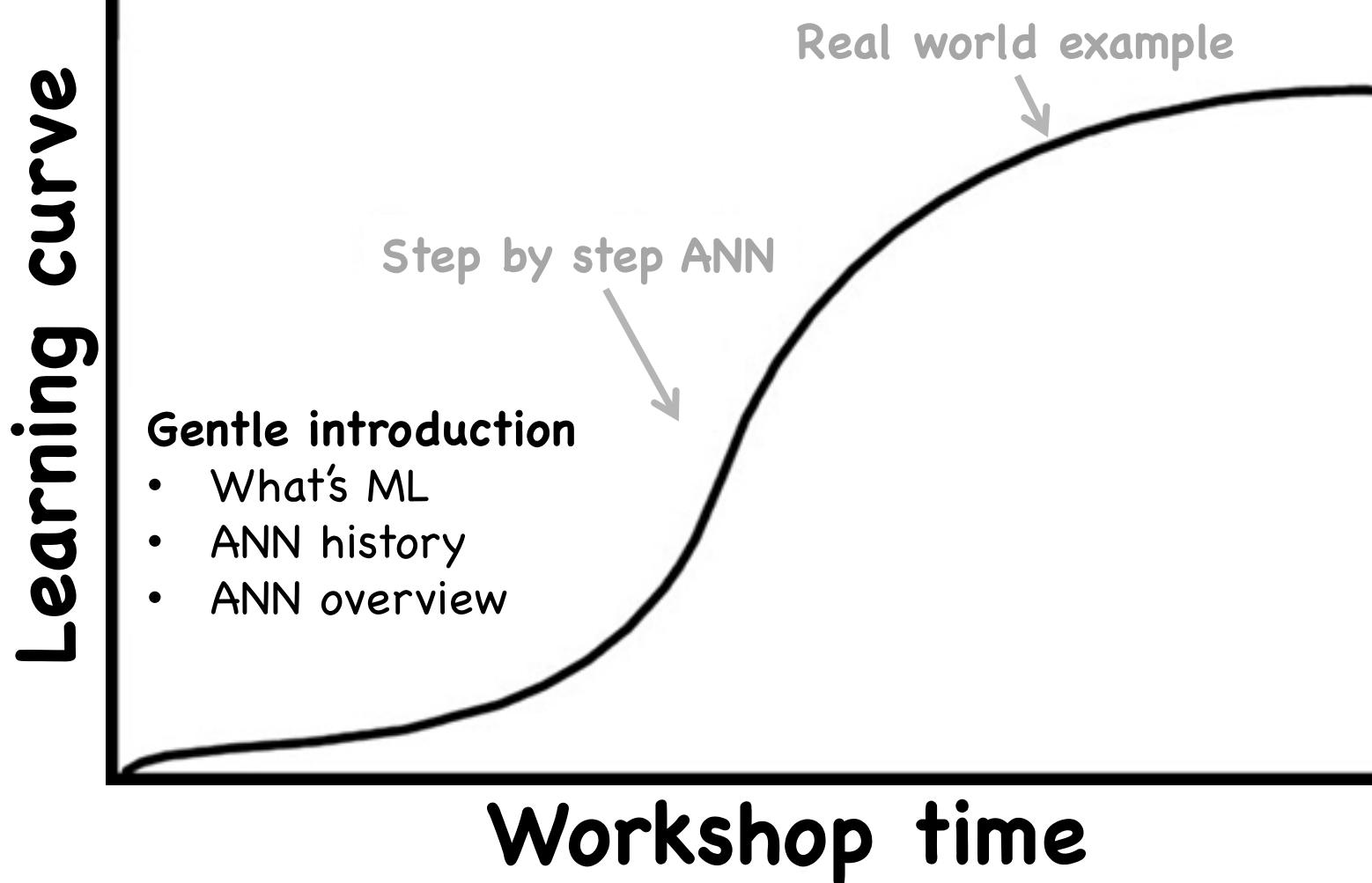
<http://seismo.berkeley.edu/qingkaikong/>



“Hey, my sensors detect that you are
scanning my cards!”

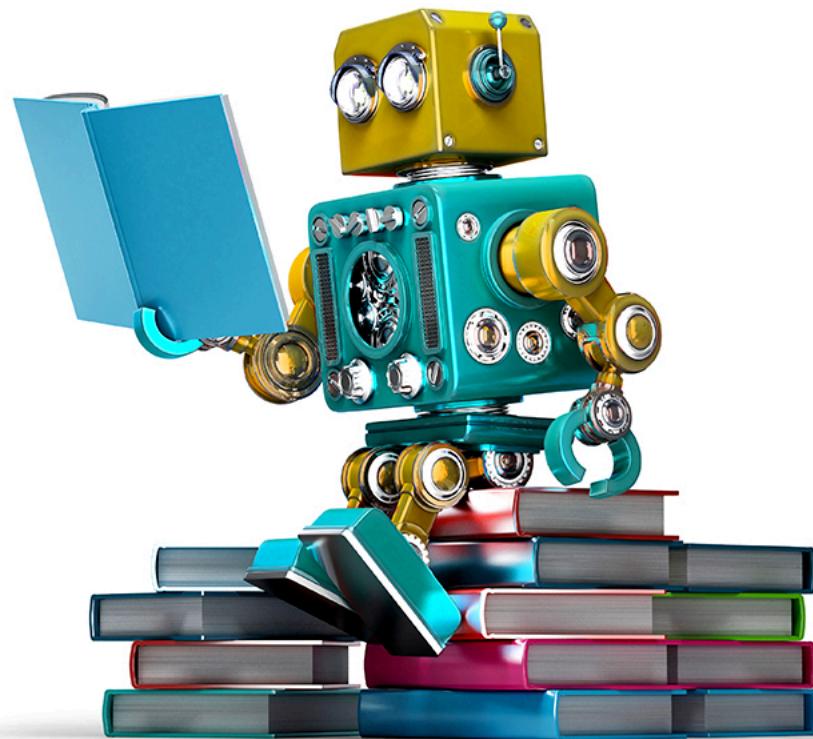


https://github.com/qingkaikong/20161202_ANN_basics



https://github.com/qingkaikong/20161202_ANN_basics

What is machine learning?



https://github.com/qingkaikong/20161202_ANN_basics



amazon.com

Recommended for You

Amazon.com has new recommendations for you based on items you purchased or told us you own.

 Google Apps Deciphered: Compute in the Cloud to Streamline Your Desktop	 Google Apps Administrator Guide: A Private-Label Web Workspace	 Googlepedia: The Ultimate Google Resource (3rd Edition)
---	--	---

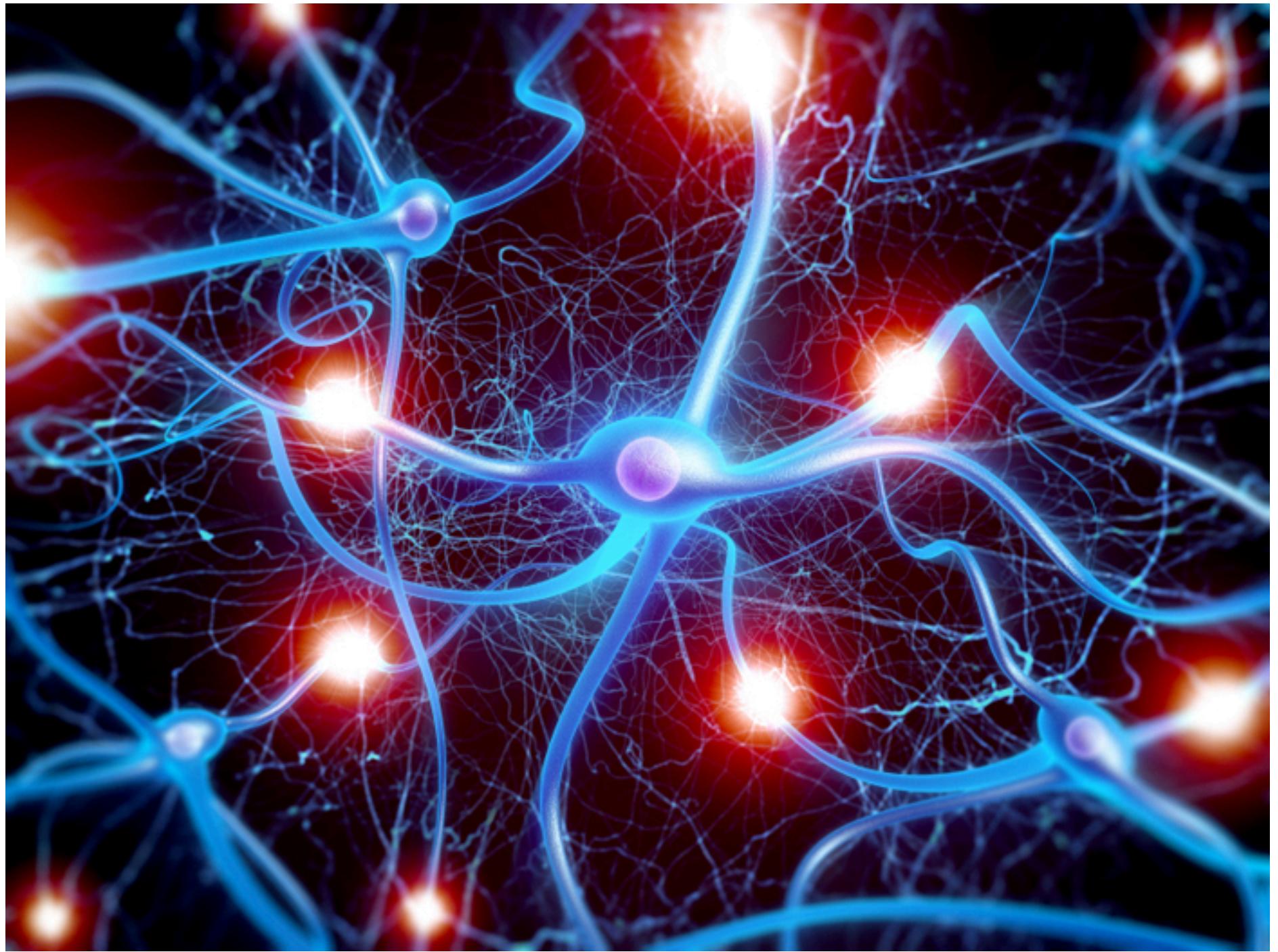
**Self-driving car
Voice recognition**

...

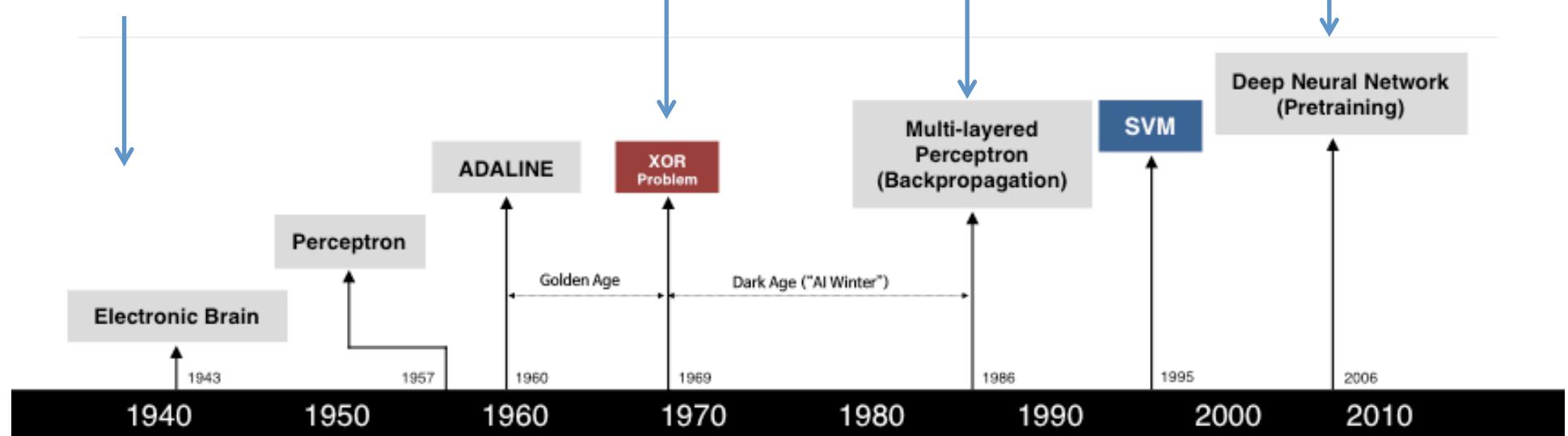
<https://github.com/qingkaikong/20161202 ANN basics>

Not always
working





1940s Birth



S. McCulloch - W. Pitts



- Adjustable Weights
- Weights are not Learned



F. Rosenblatt



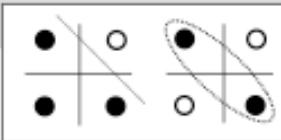
B. Widrow - M. Hoff



- Learnable Weights and Threshold



M. Minsky - S. Papert



• XOR Problem



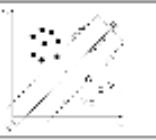
D. Rumelhart - G. Hinton - R. Williams



- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting



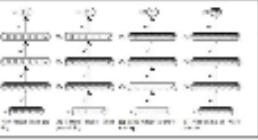
V. Vapnik - C. Cortes



- Limitations of learning prior knowledge
- Kernel function: Human Intervention

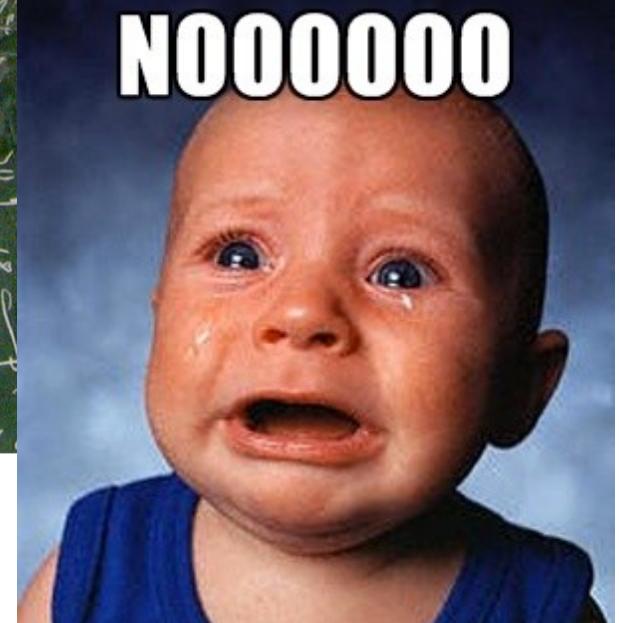


G. Hinton - S. Ruslan



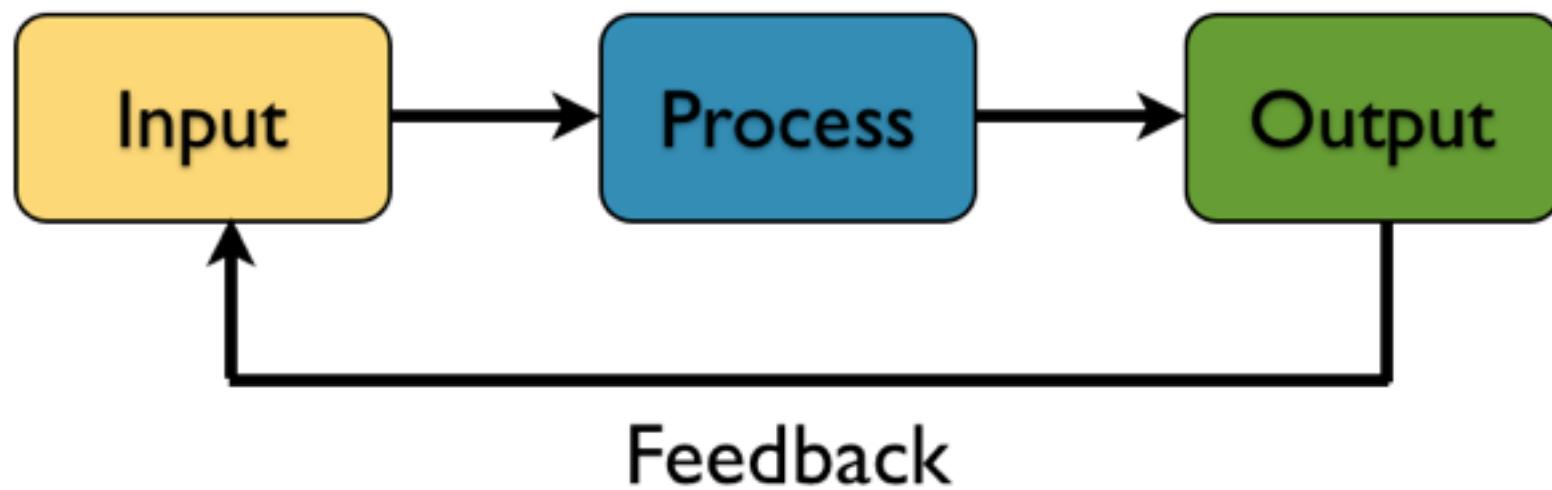
- Hierarchical feature Learning

The image shows a blackboard covered in mathematical notation, including various symbols, equations, and diagrams. In the bottom right corner, there is a large, distressed image of a baby's face with the word "NOOOOO" written in large, bold, white letters across it.

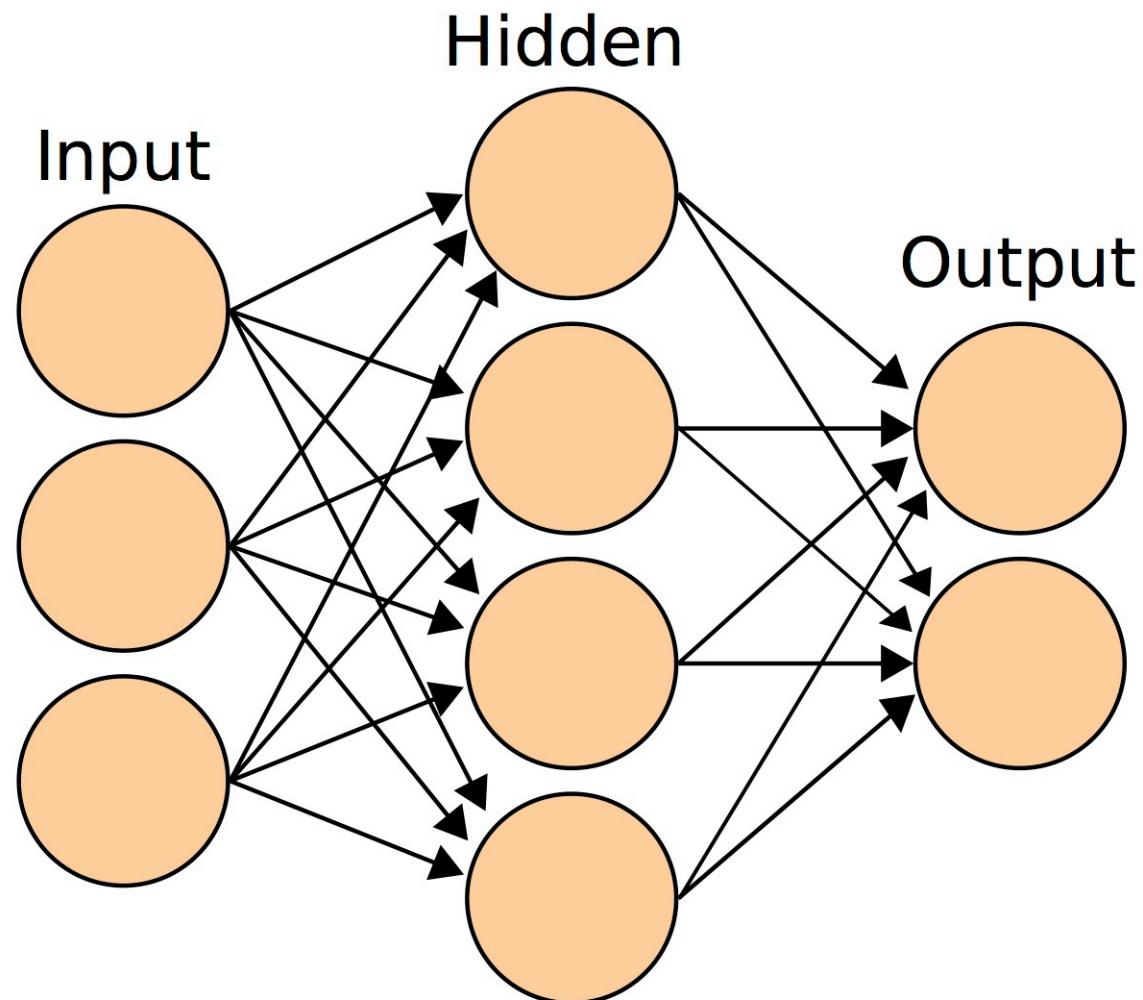




ANN in simple view



ANN jargons



What're the weights





YOU'RE IN MY SPOT

GRAPHICS GARAGE

Input



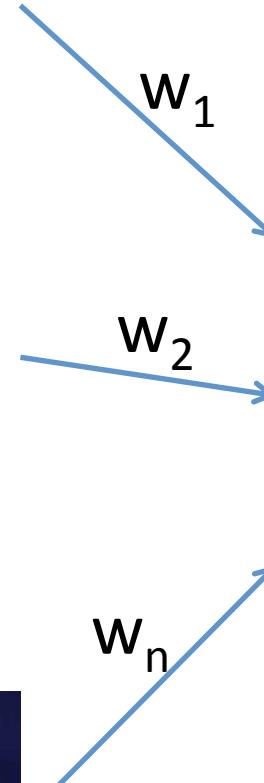
.

.

.



Intuitive Artificial Neural Network



$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$

Output



Input



•
•
•



Intuitive Artificial Neural Network

Output



$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$

error
feedback



Input



•
•
•



Intuitive Artificial Neural Network

Output



$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$



error
feedback



Input



.

.

.



Intuitive Artificial Neural Network

Output

$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$



w_1

w_2

w_n



Learning curve

Workshop time

Gentle introduction

- What's ML
- ANN history
- ANN overview

Step by step ANN

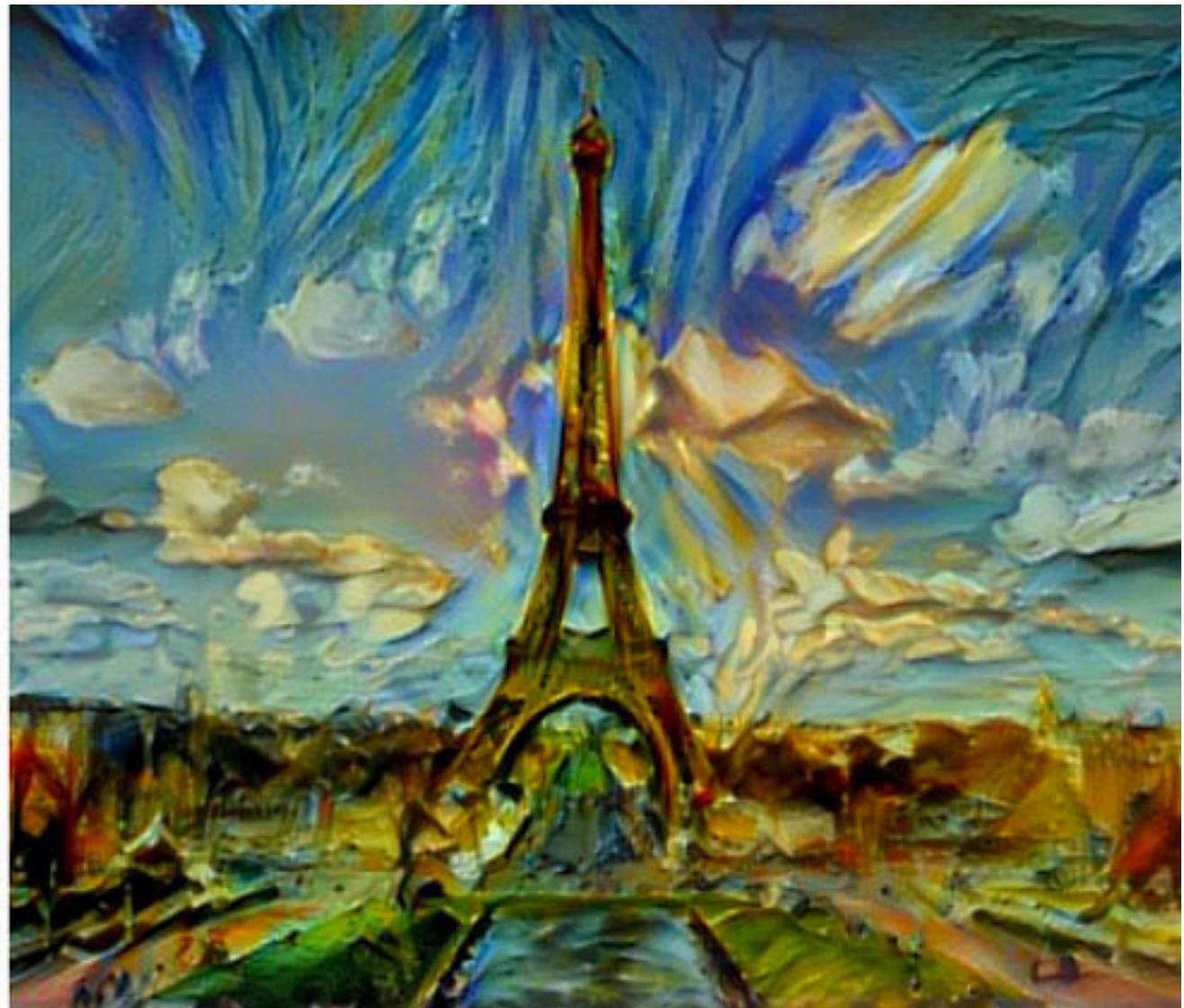
- Perceptron
- Backpropagation

Real world example

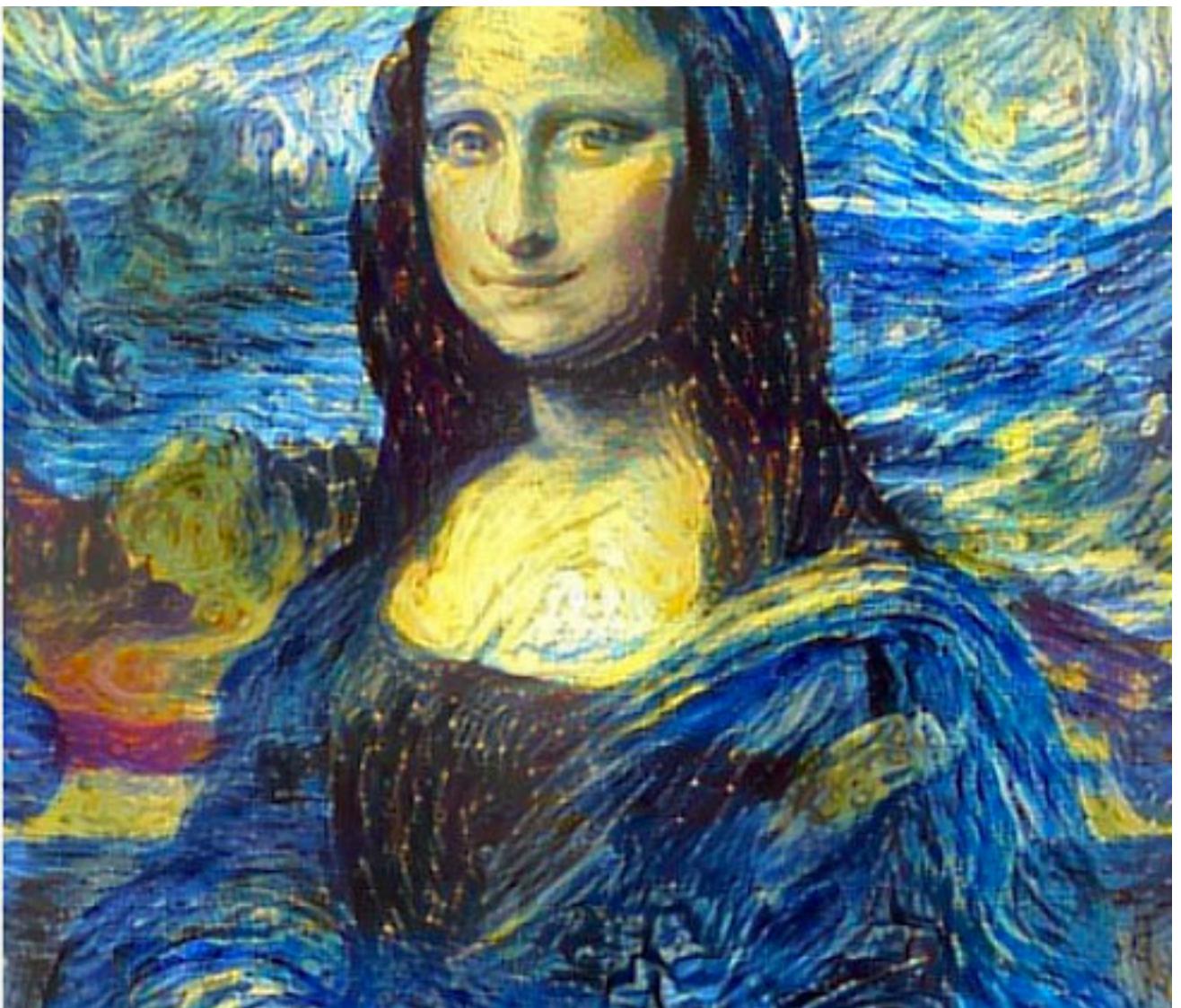




Application: Learn arts

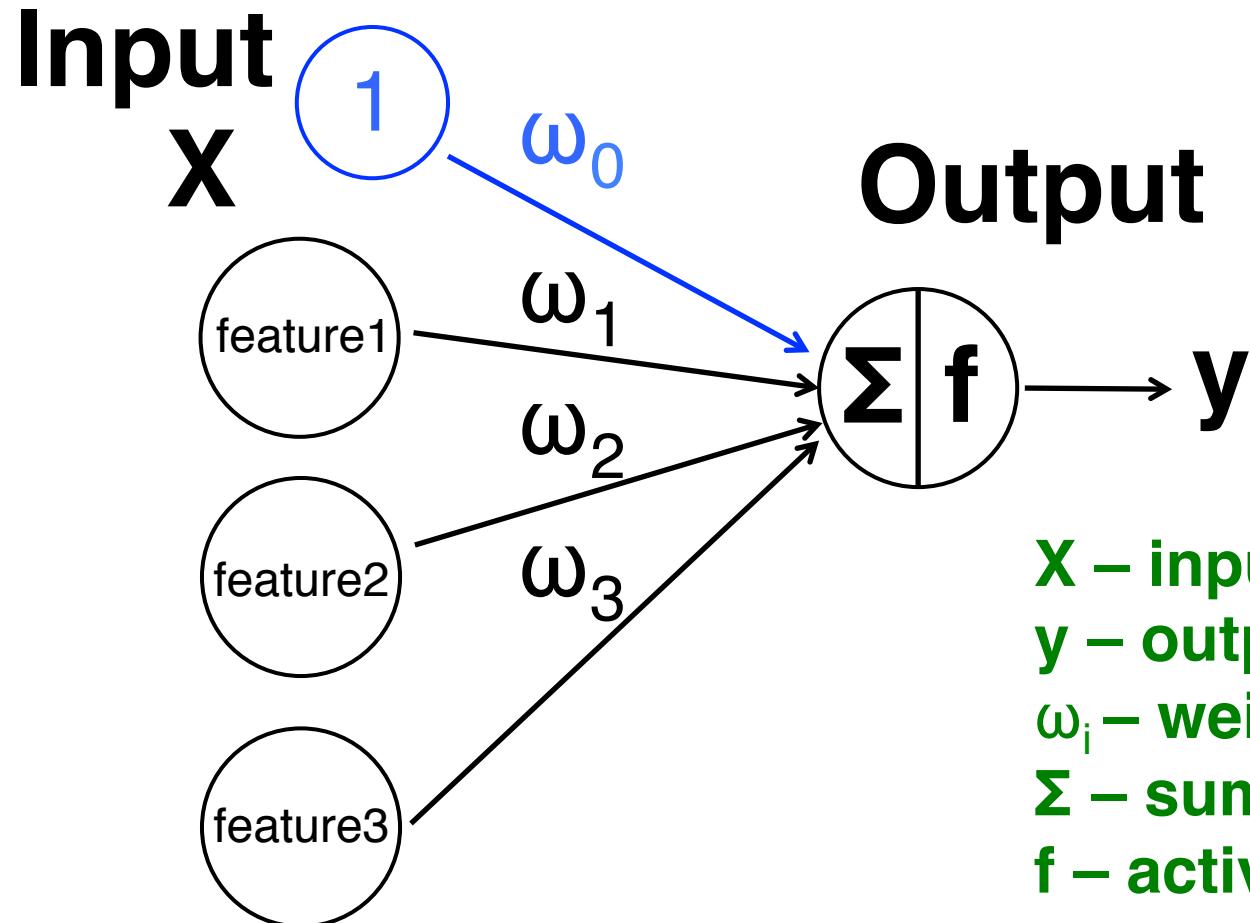


<https://arxiv.org/abs/1508.06576v1>
<https://deepoch.io/>



<http://junkhost.com/2016/03/man-combines-random-peoples-photos-using-neural-networks-and-the-results-are-amazing/>



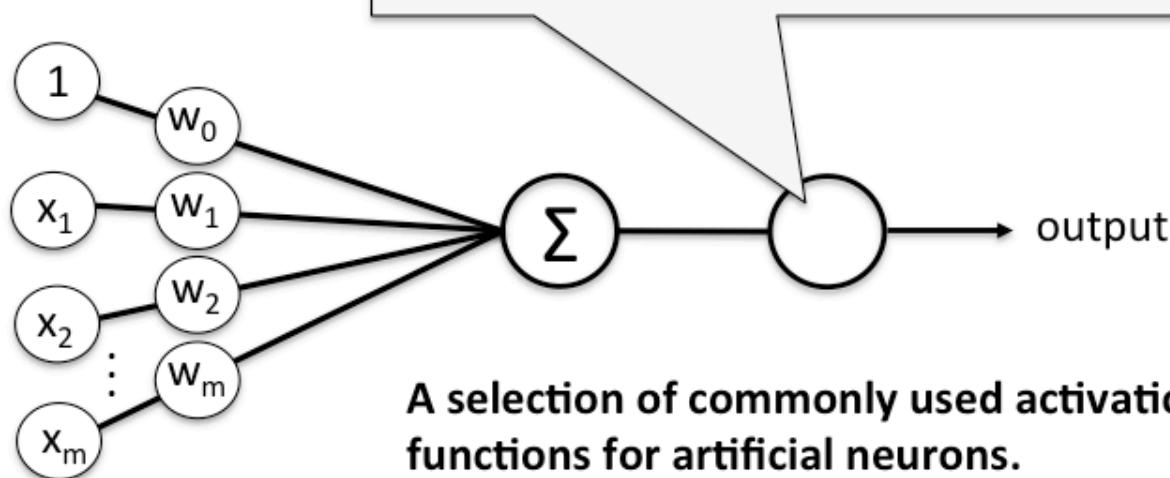


X – input data
y – output target
 ω_i – weights
 Σ – summation
f – activation function
Blue circle – bias

$$\Sigma = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n$$

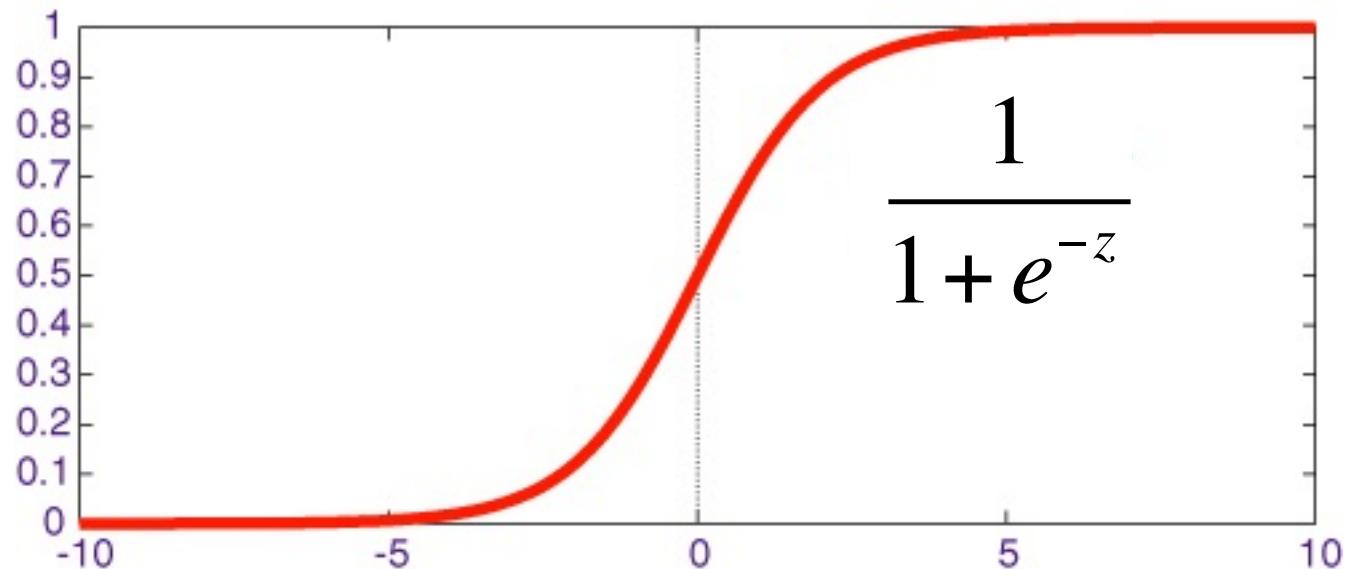
$$f = f(\omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n)$$

	Unit step	$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise.} \end{cases}$
		$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise.} \end{cases}$
	Linear	$g(z) = z$
	Logistic (sigmoid)	$g(z) = 1 / (1 + \exp(-z))$
	Hyperbolic tangent (sigmoid)	$g(z) = \frac{\exp(2z) - 1}{\exp(2z) + 1}$
...		



A selection of commonly used activation functions for artificial neurons.

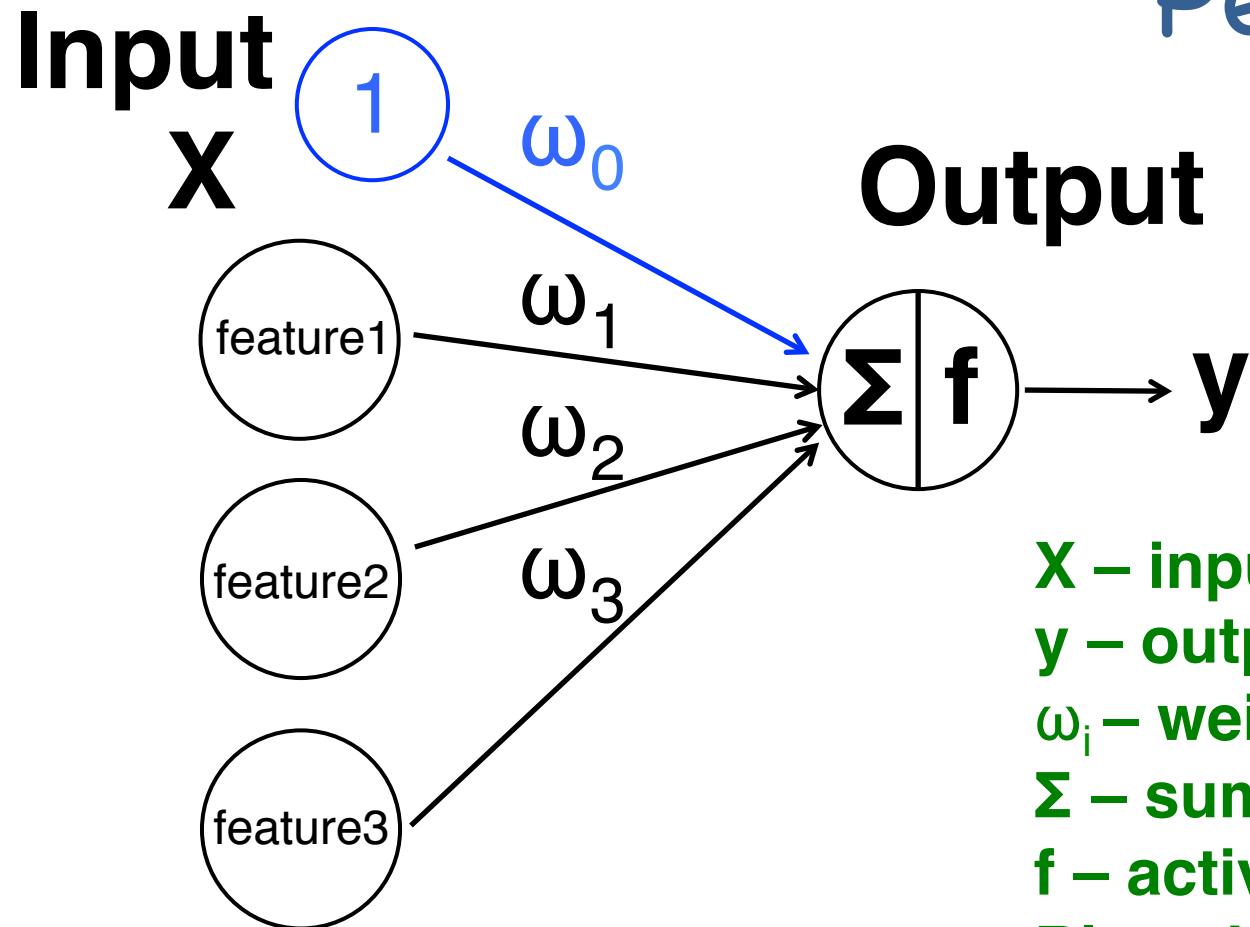
More activation function



$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n$$

$$f(z) = \frac{1}{1 + e^{-z}} \quad \frac{df(z)}{dx} = f(z)(1 - f(z))$$

Perceptron



X – input data
y – output target

ω_i – weights

Σ – summation

f – activation function

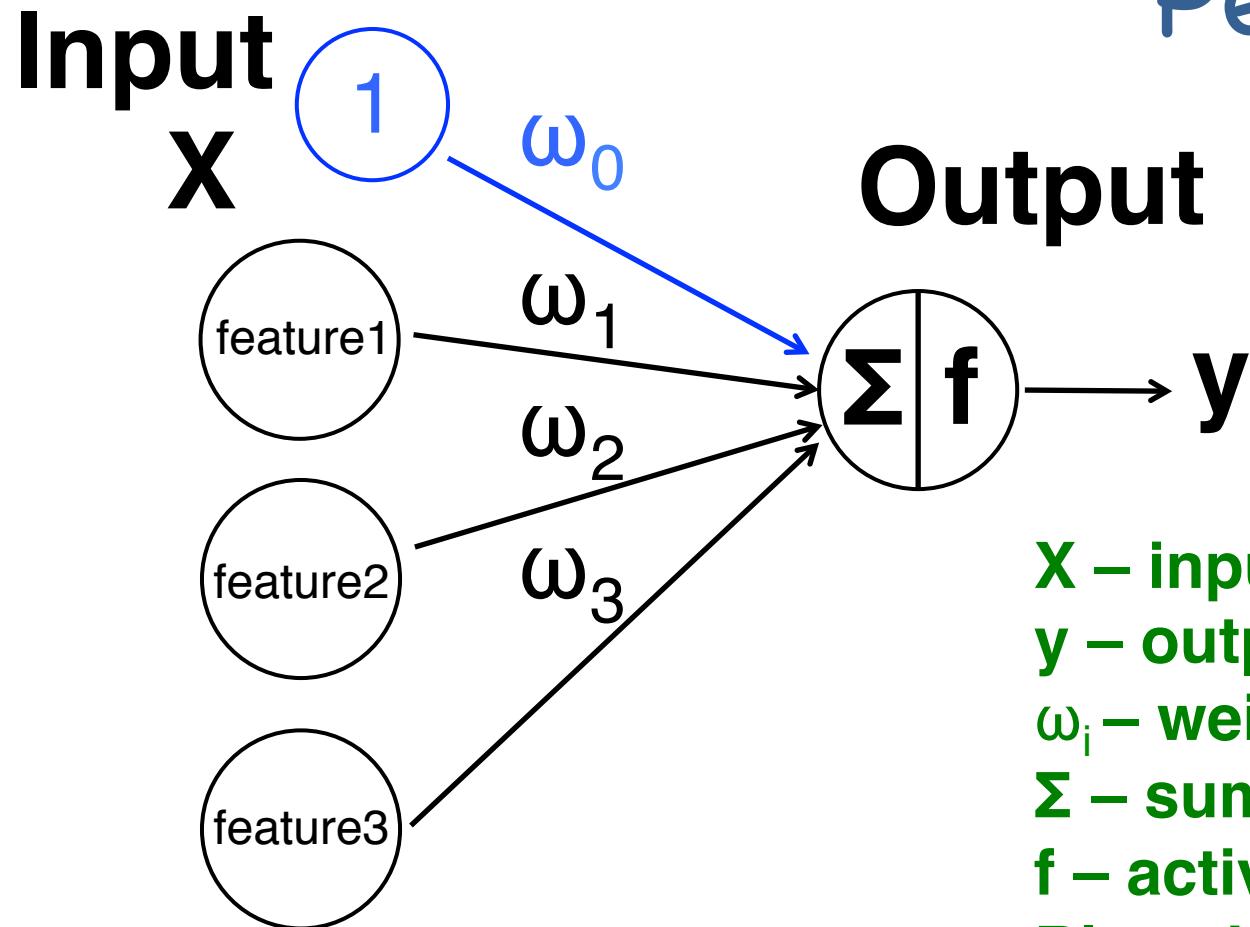
Blue circle – bias

$$\Sigma = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n$$

$$f = f(\omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n)$$

y This is our estimation

Perceptron



X – input data
y – output target
 ω_i – weights
 Σ – summation
f – activation function
Blue circle – bias

Error = Target - Estimation

~~Error~~



How the ANN learns

Error = Target - Estimation

How the ANN learns

Error = Target - Estimation

Learning:

Update weights to reduce error next time!



HOW?

Weights update rules

Weights Delta = Error × slope × input

How much we will update
the weights for next time



Look at errors closer

Error = Target - Estimation

Target



Look at errors closer

Error = Target - Estimation

Target



Three cases:

- Error < 0: Target is 0, estimation is not 0
- Error > 0: Target is 1, estimation is not 1
- Error = 0: Estimation correct

Look at errors closer
(assume inputs are positive)

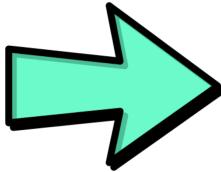
Cases 1:



- Error < 0: Target is **0**, estimation is not 0

Look at errors closer
(assume inputs are positive)

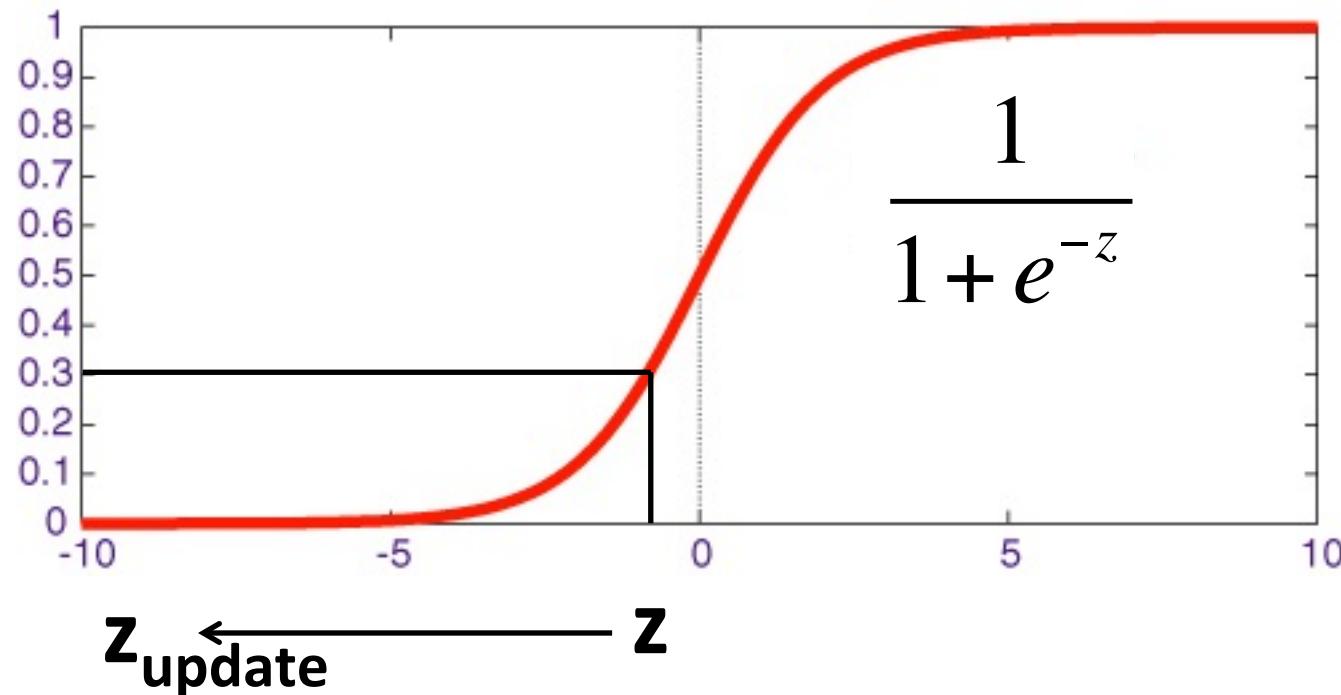
Cases 1:

- Target is 0
 - Estimation is 0.3
- 
- Error = $0 - 0.3 = -0.3$

Look at errors closer
(assume inputs are positive)

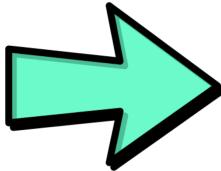
Cases 1:

- Target is 0
 - Estimation is 0.3
- Error = $0 - 0.3 = -0.3$



Look at errors closer
(assume inputs are positive)

Cases 1:

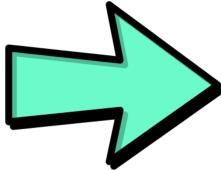
- Target is 0
 - Estimation is 0.3
- 
- Error = $0 - 0.3 = -0.3$

$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

We need **reduce weights!**

Look at errors closer
(assume inputs are positive)

Cases 1:

- Target is 0
 - Estimation is 0.3
- 
- Error = $0 - 0.3 = -0.3$

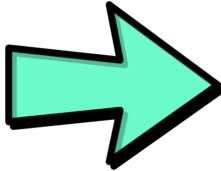
$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

We need **reduce weights!**

If we add error to the weights, we will reduce it!

Look at errors closer
(assume inputs are positive)

Cases 1:

- Target is 0
 - Estimation is 0.3
- 
- Error = $0 - 0.3 = -0.3$

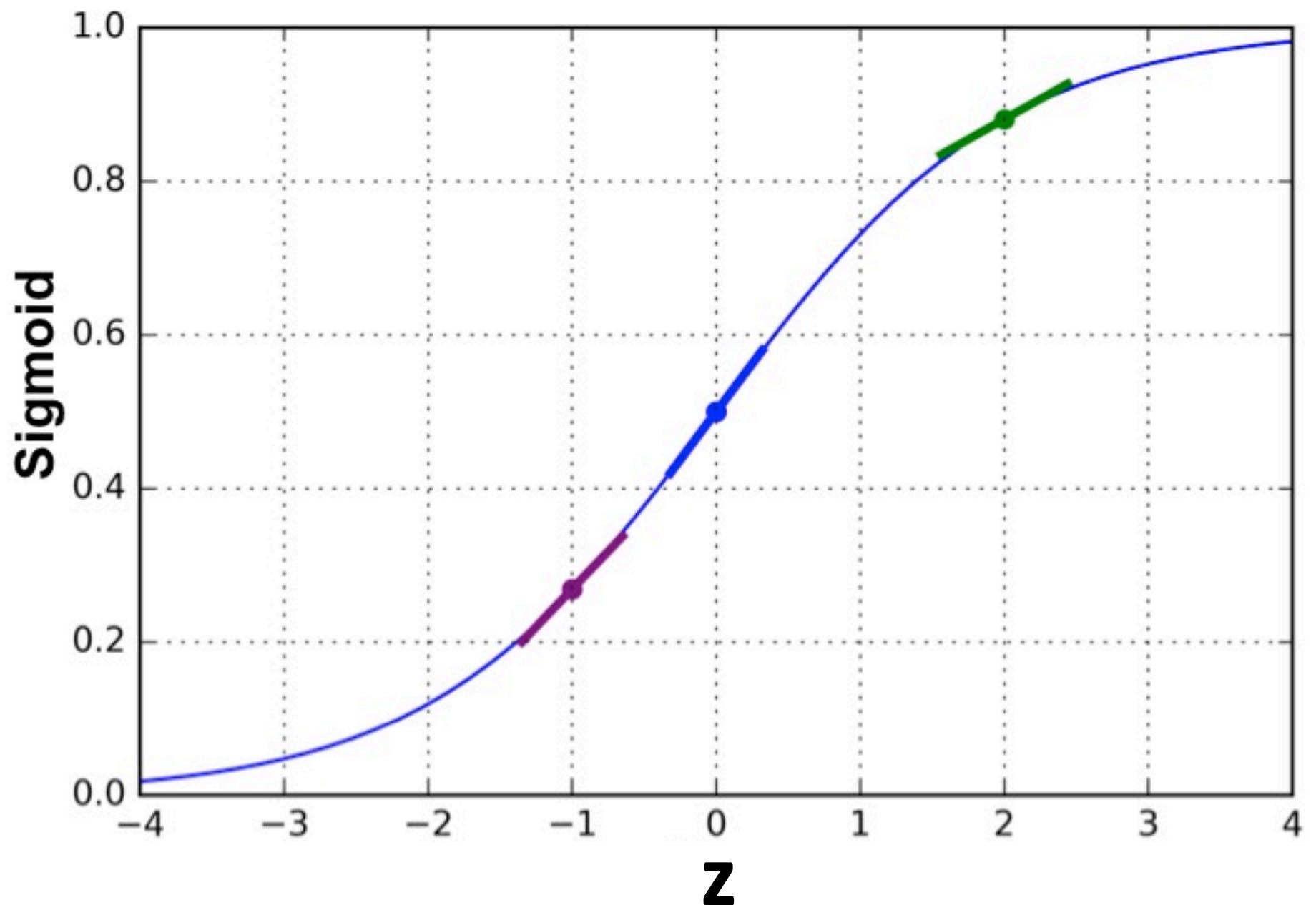
$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

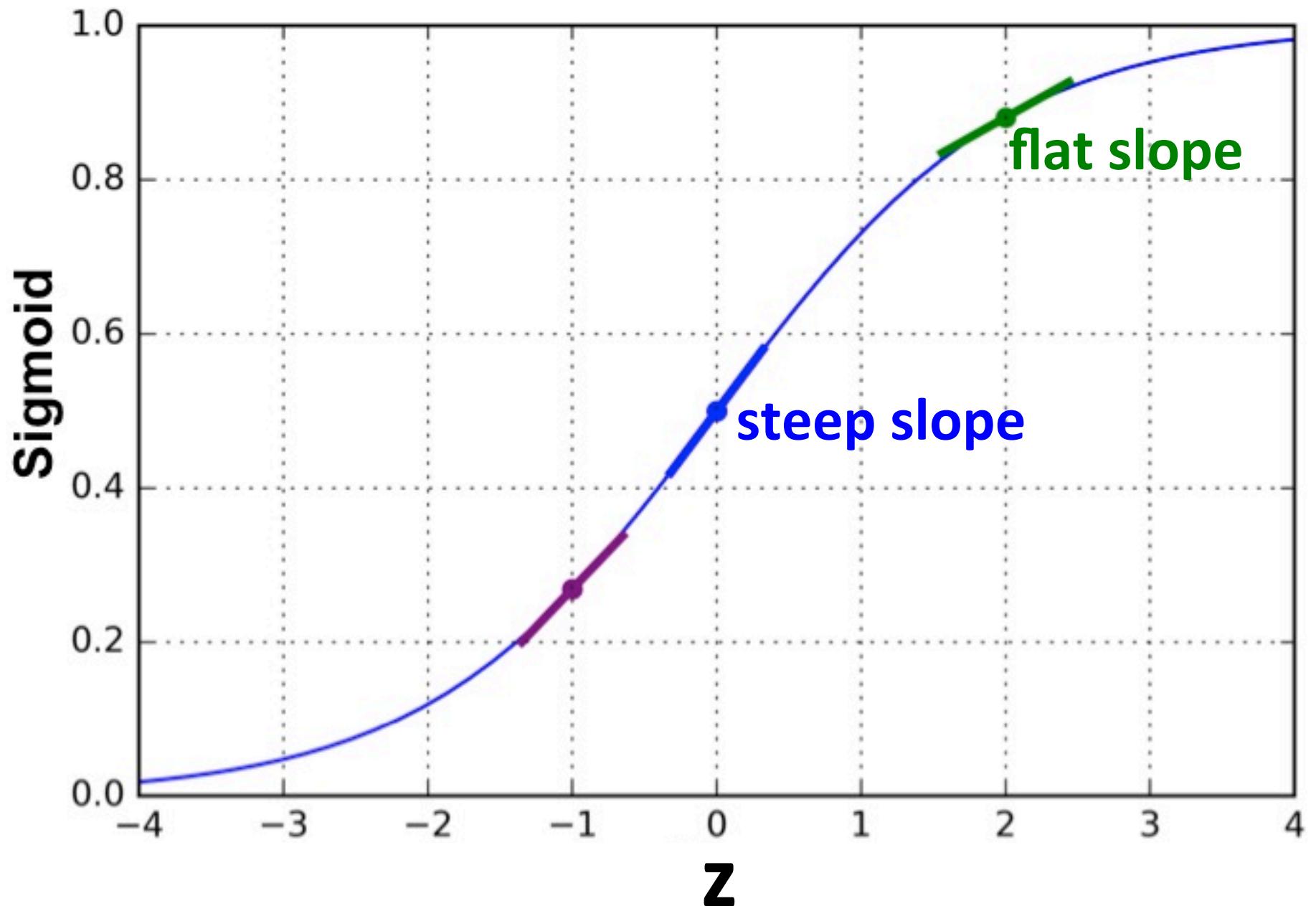
We need **reduce weights!**

But what if the inputs are negative

Weights update rules

Weights Delta = Error × input



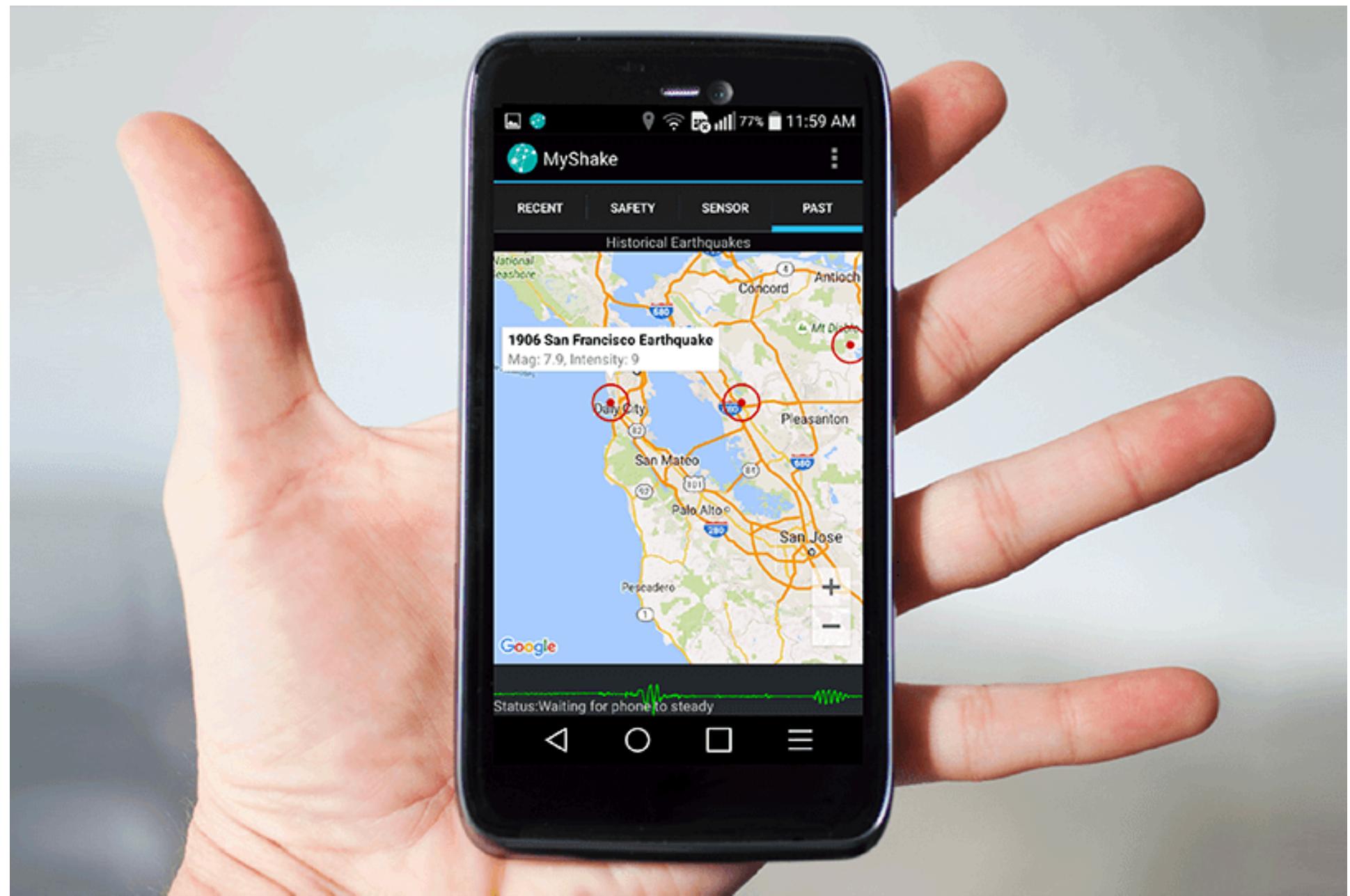


Weights update rules

Weights Delta = Error × slope × input

Learn from example



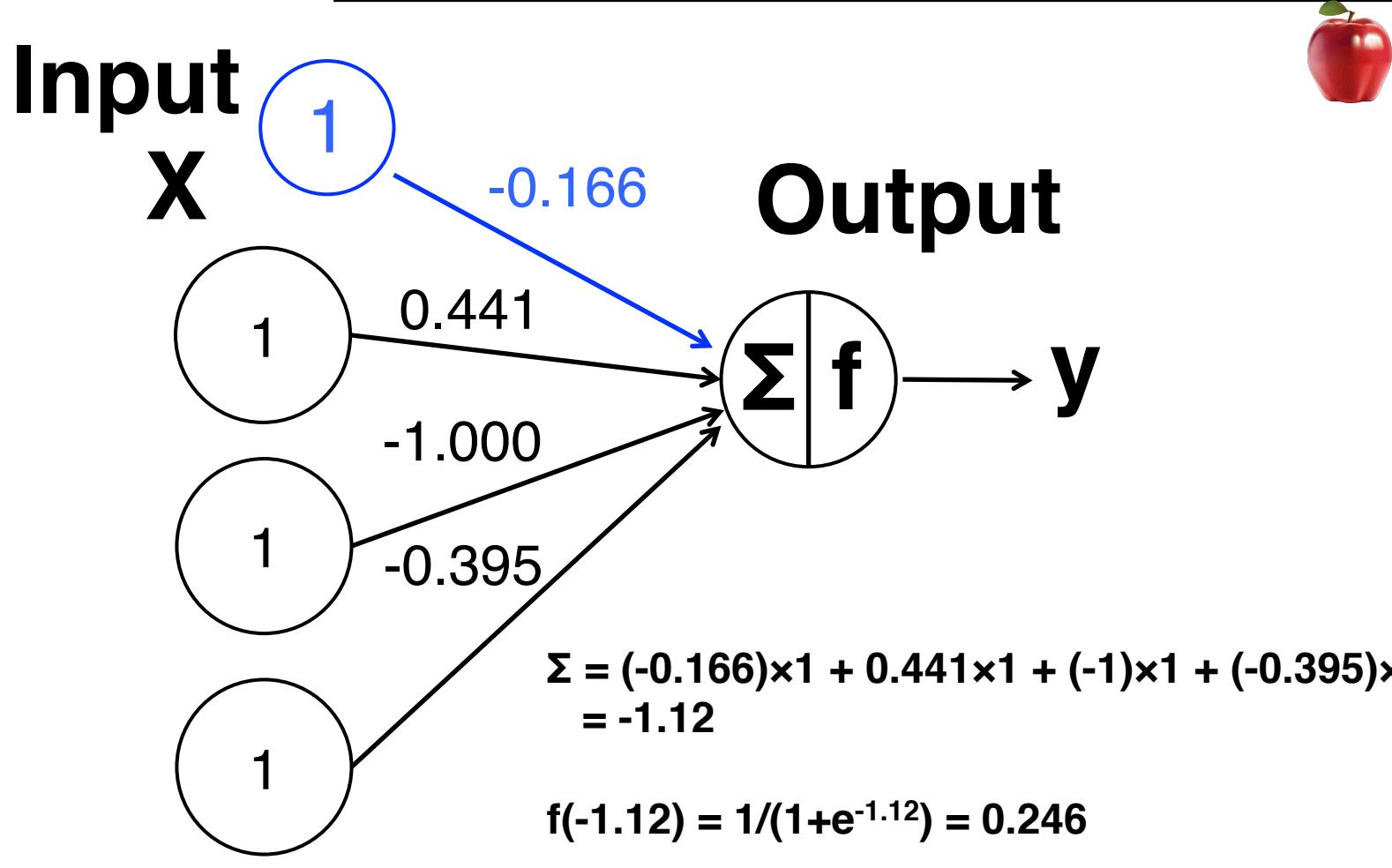


Learn from Example

Sample	Feature 1	Feature 2	Feature 3	Target
Sample 1	0	0	1	0 
Sample 2	1	1	1	1 
Sample 3	1	0	1	1 
Sample 4	0	1	1	0 
Sample 5	1	0	0	1 

How to deal with errors

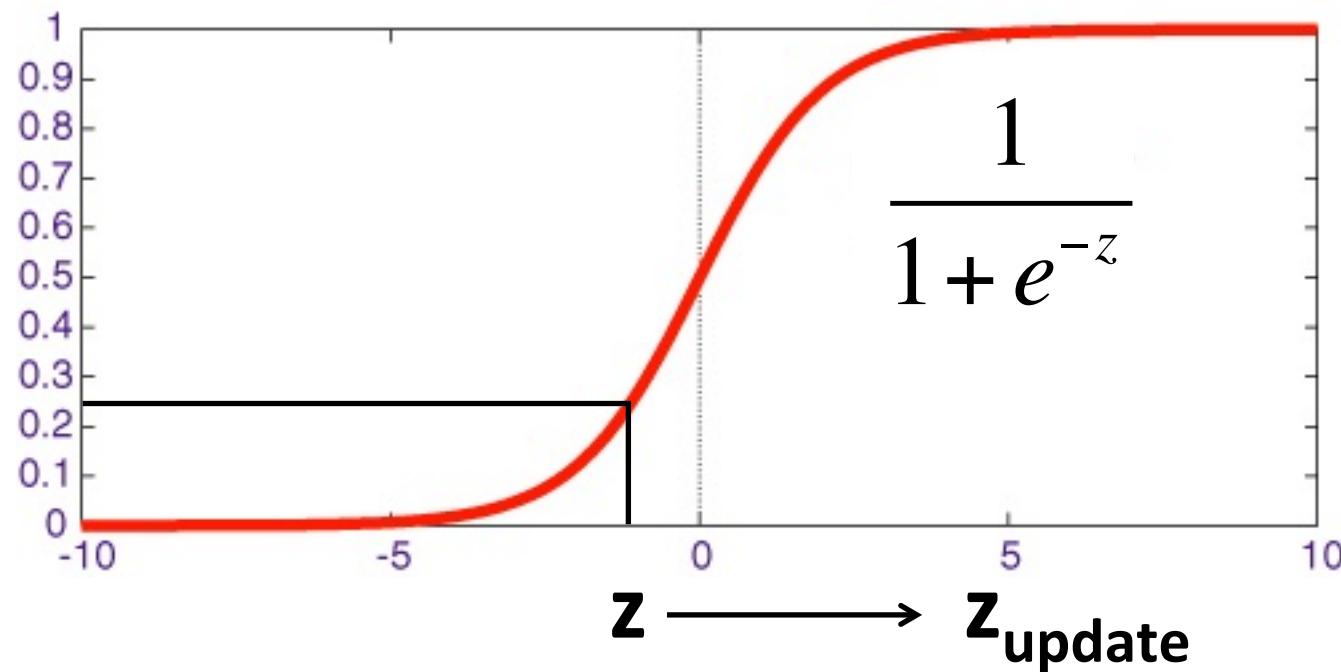
Sample	Feature 1	Feature 2	Feature 3	Target
Sample 1	1	1	1	1



- Target: 1
- Estimation: 0.246



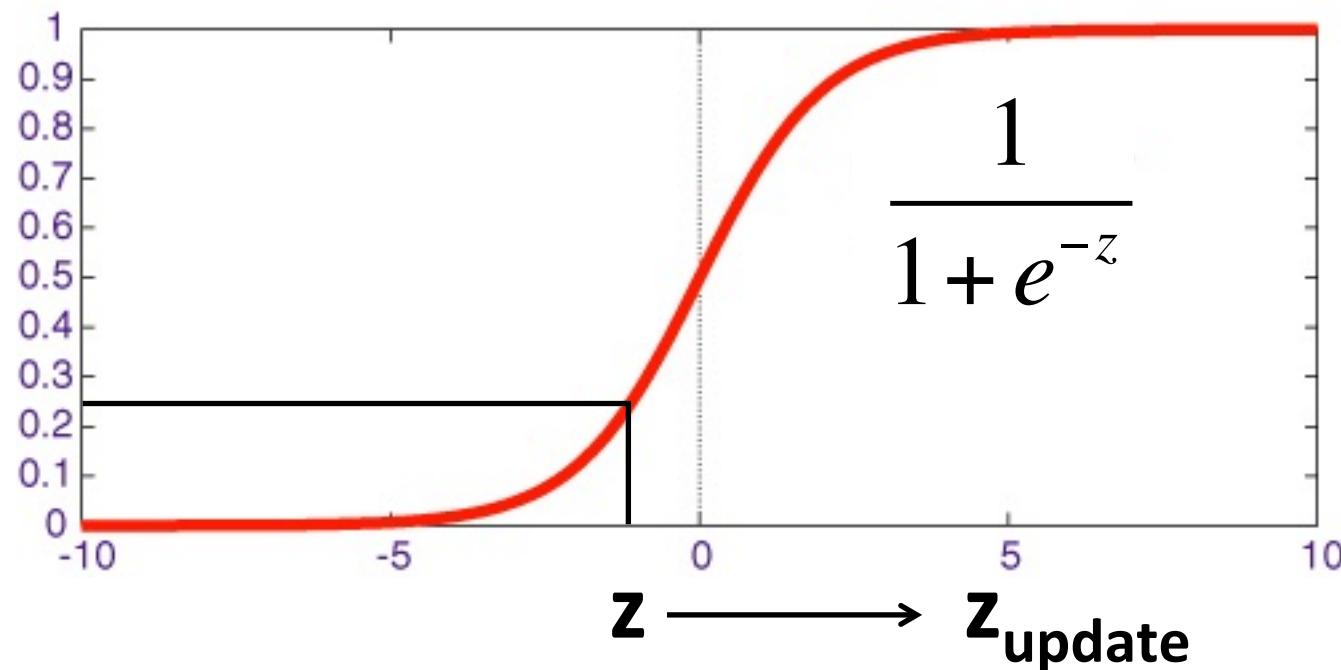
Error 0.754



- Target: 1
- Estimation: 0.246

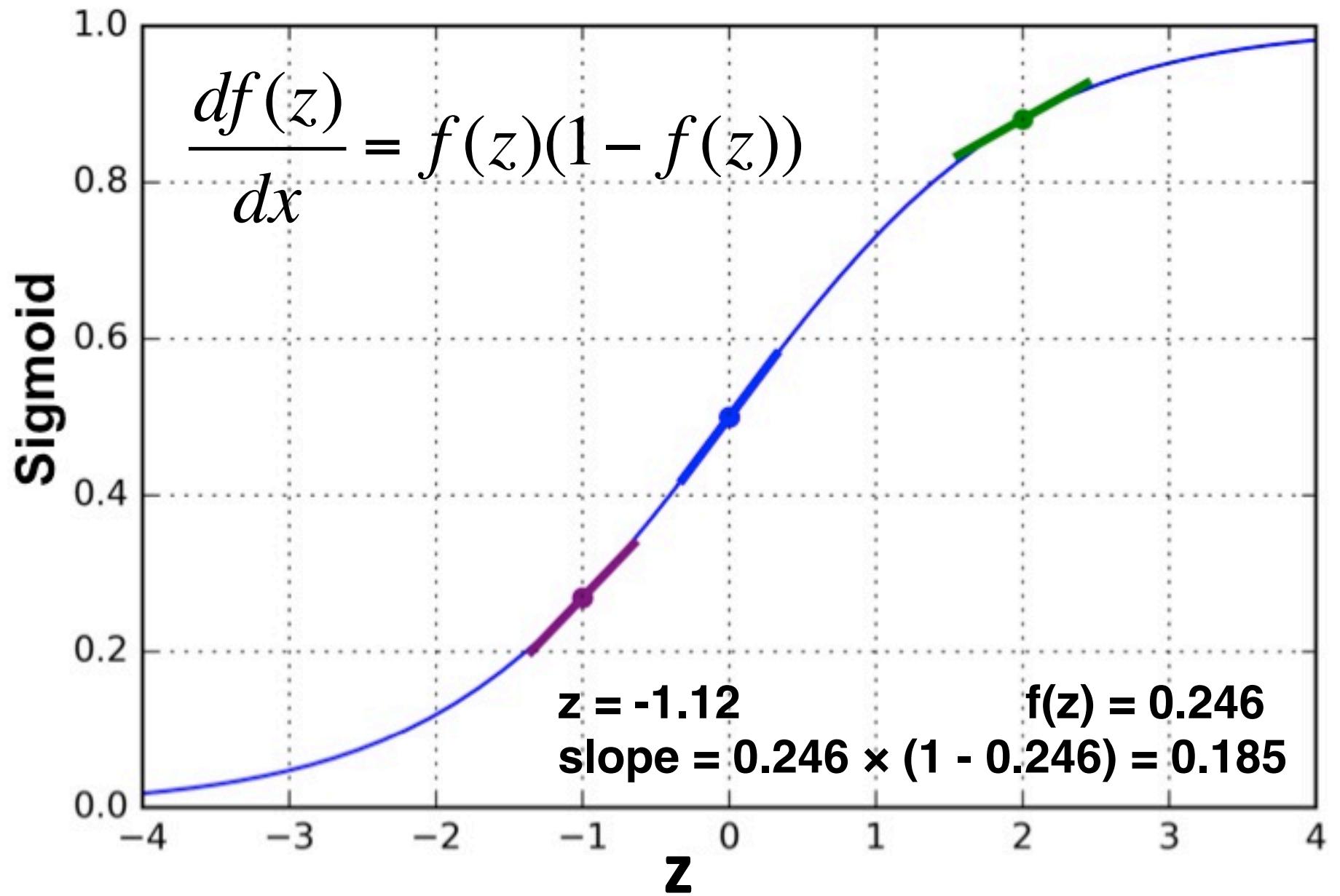


Error 0.754



We want to **increase** the weights next time to have larger z

Weights delta = 0.754 × slope × input



$$\begin{aligned}\text{Change item} &= \color{red}{0.754} \times \color{green}{0.185} \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.139 \\ 0.139 \\ 0.139 \\ 0.139 \\ 0.139 \end{bmatrix}\end{aligned}$$

Changes of the error

$$\begin{aligned} \text{Updated Weights} &= \begin{bmatrix} -0.166 \\ 0.441 \\ -1.000 \\ -0.395 \end{bmatrix} + \begin{bmatrix} 0.139 \\ 0.139 \\ 0.139 \\ 0.139 \end{bmatrix} = \begin{bmatrix} -0.027 \\ 0.580 \\ -0.861 \\ -0.256 \end{bmatrix} \\ &\quad \begin{array}{c} \nearrow \\ \text{Original Weights} \end{array} \qquad \begin{array}{c} \nearrow \\ \text{updates} \end{array} \end{aligned}$$

Changes of the error

$$\begin{array}{l} \text{Updated} \\ \text{Weights} \end{array} = \begin{bmatrix} -0.166 \\ 0.441 \\ -1.000 \\ -0.395 \end{bmatrix} + \begin{bmatrix} 0.139 \\ 0.139 \\ 0.139 \\ 0.139 \end{bmatrix} = \begin{bmatrix} -0.027 \\ 0.580 \\ -0.861 \\ -0.256 \end{bmatrix}$$

Error of next iteration

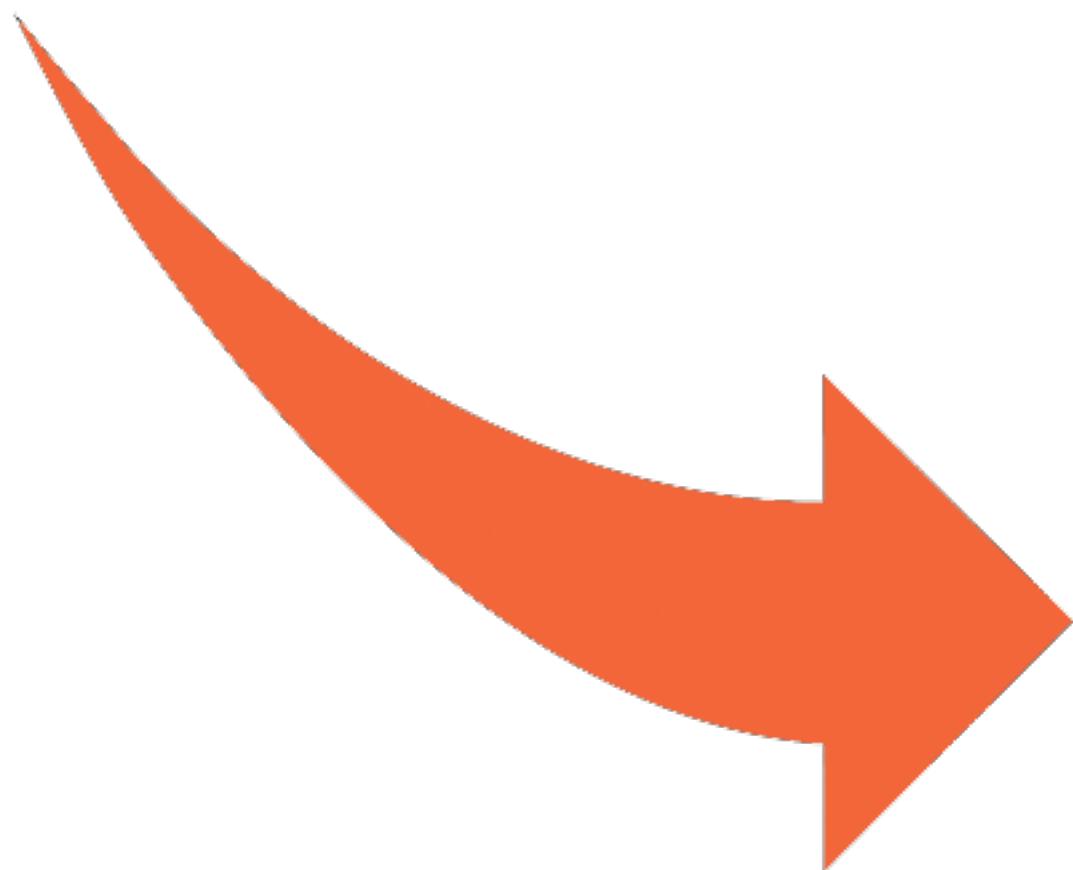
$$z = (-0.027) \times 1 + 0.580 \times 1 + (-0.861) \times 1 + (-0.256) \times 1 = -0.564$$

$$f(z) = 0.637$$

$$\text{Error} = 1 - 0.637 = \textcolor{red}{0.363}$$

Changes of the error

0.754



0.363

Iterate many times



Go to notebook 01

Application: DeepDrumpf



<https://twitter.com/DeepDrumpf>



DeepDrumpf @DeepDrumpf · Mar 9

I love the states. I win them. Ohio is beautiful, I buy it. Thank you very much. I buy Hillary, it's beautiful and I'm happy about it.



DeepDrumpf @DeepDrumpf · 18h

I'm going to be a good president of the world. Ted can't do that.



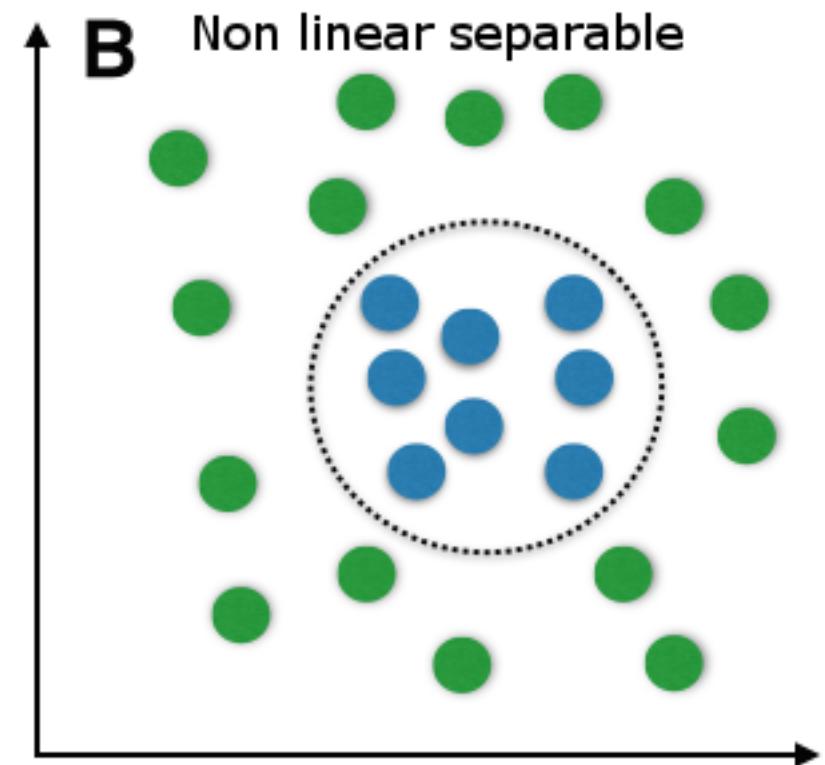
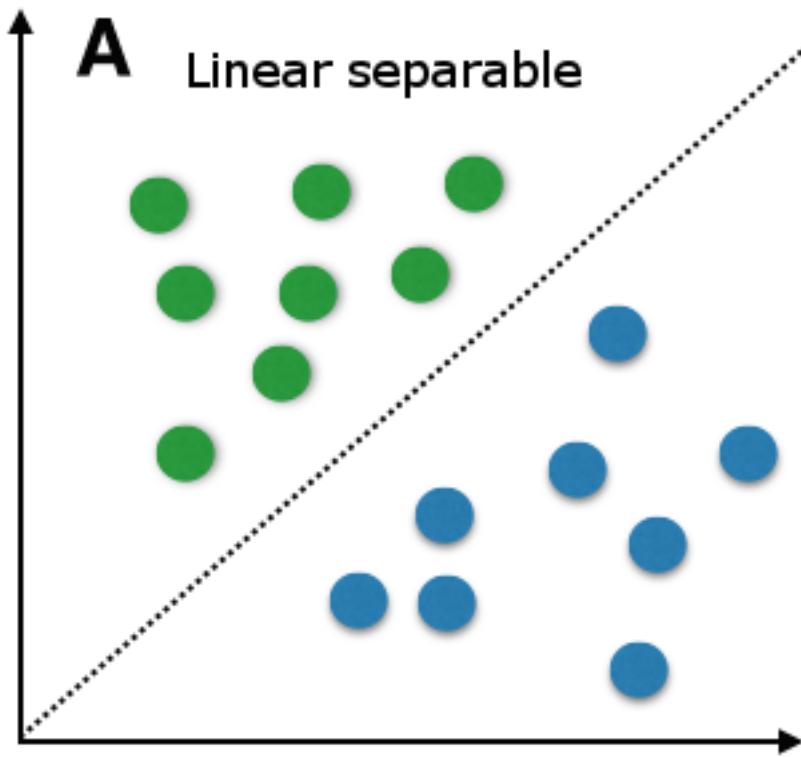
80



152

...

Perceptron limitations

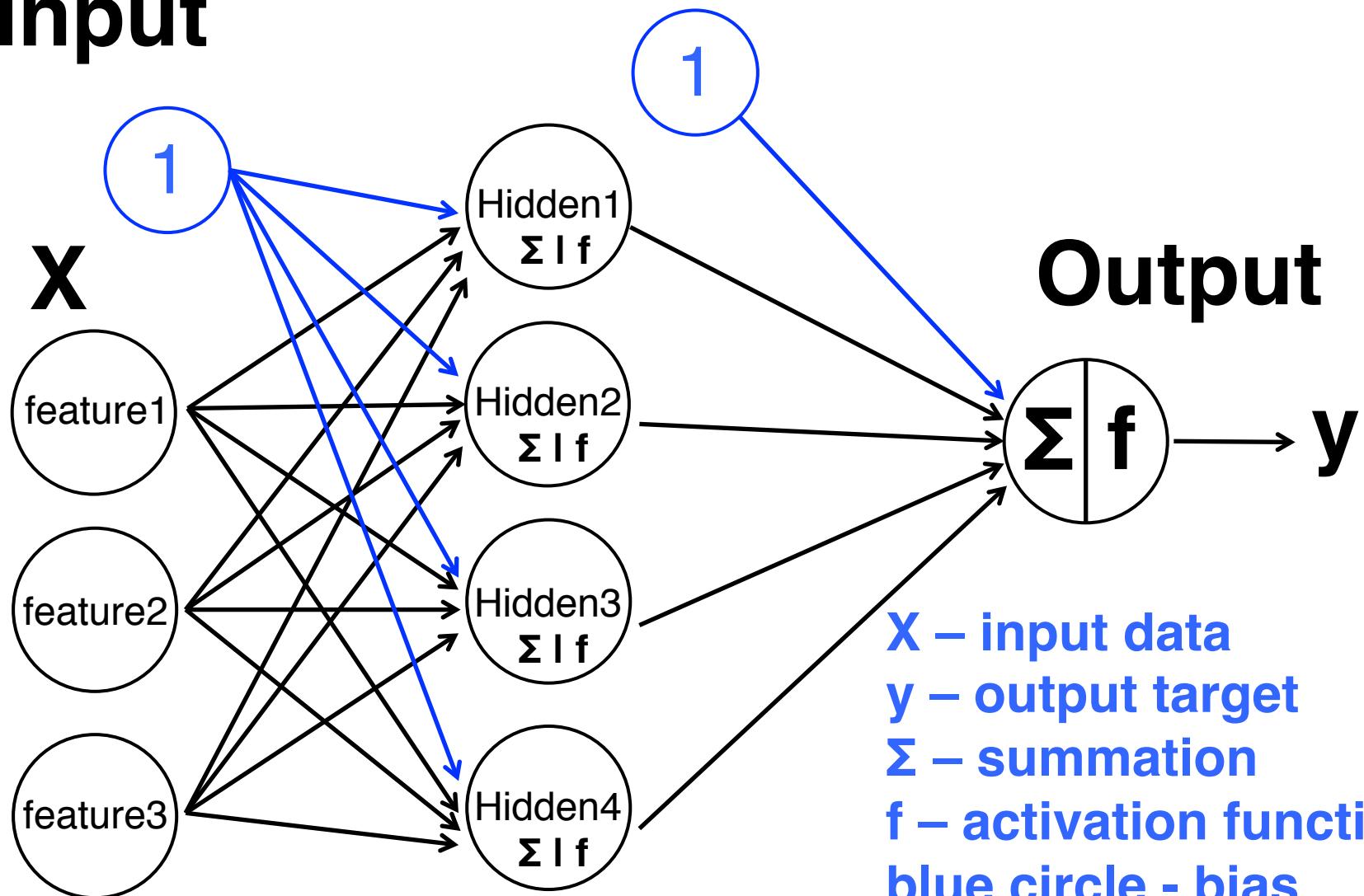


Winter of ANN



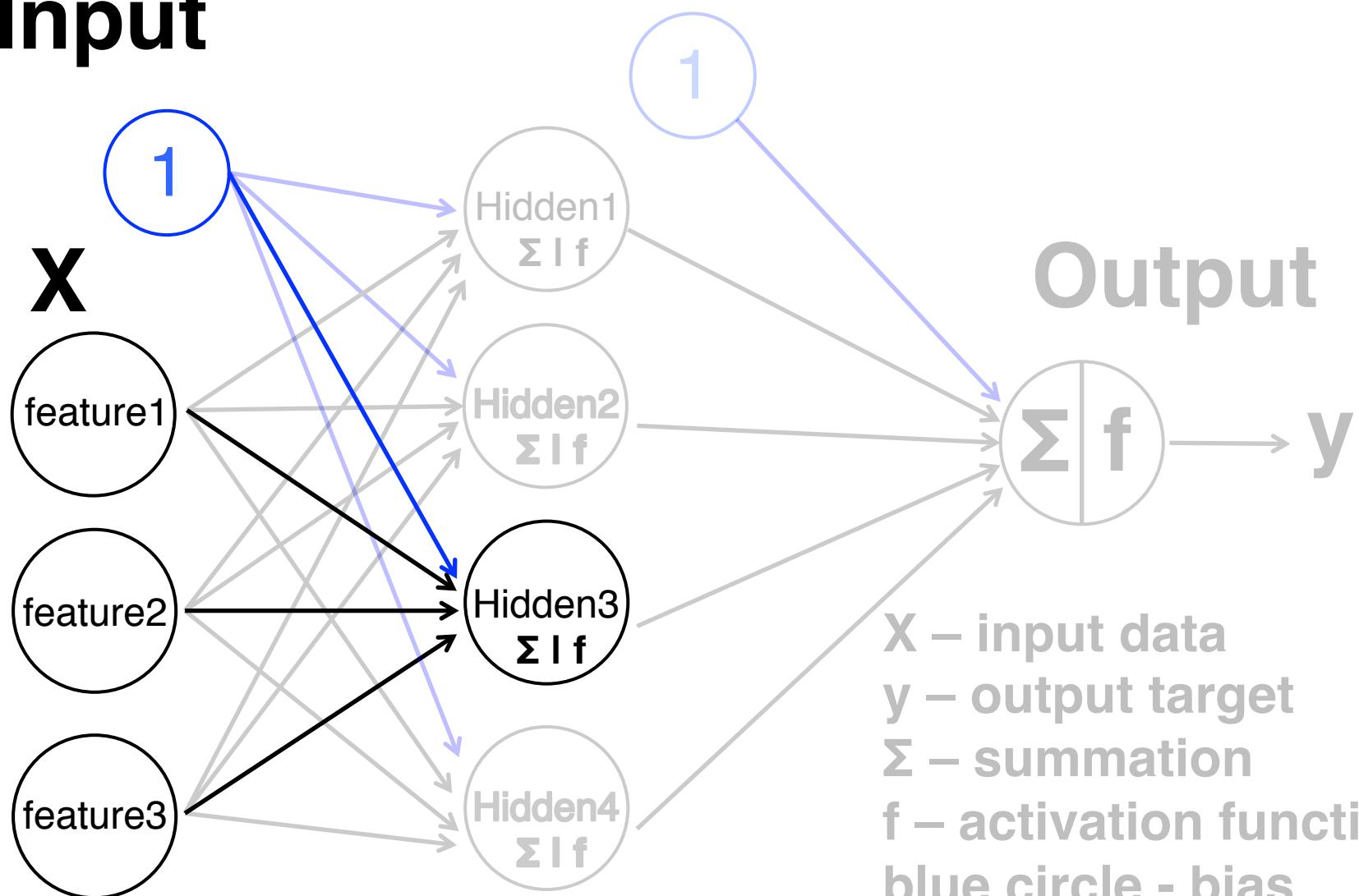
Multi-Layer Perceptron

Input



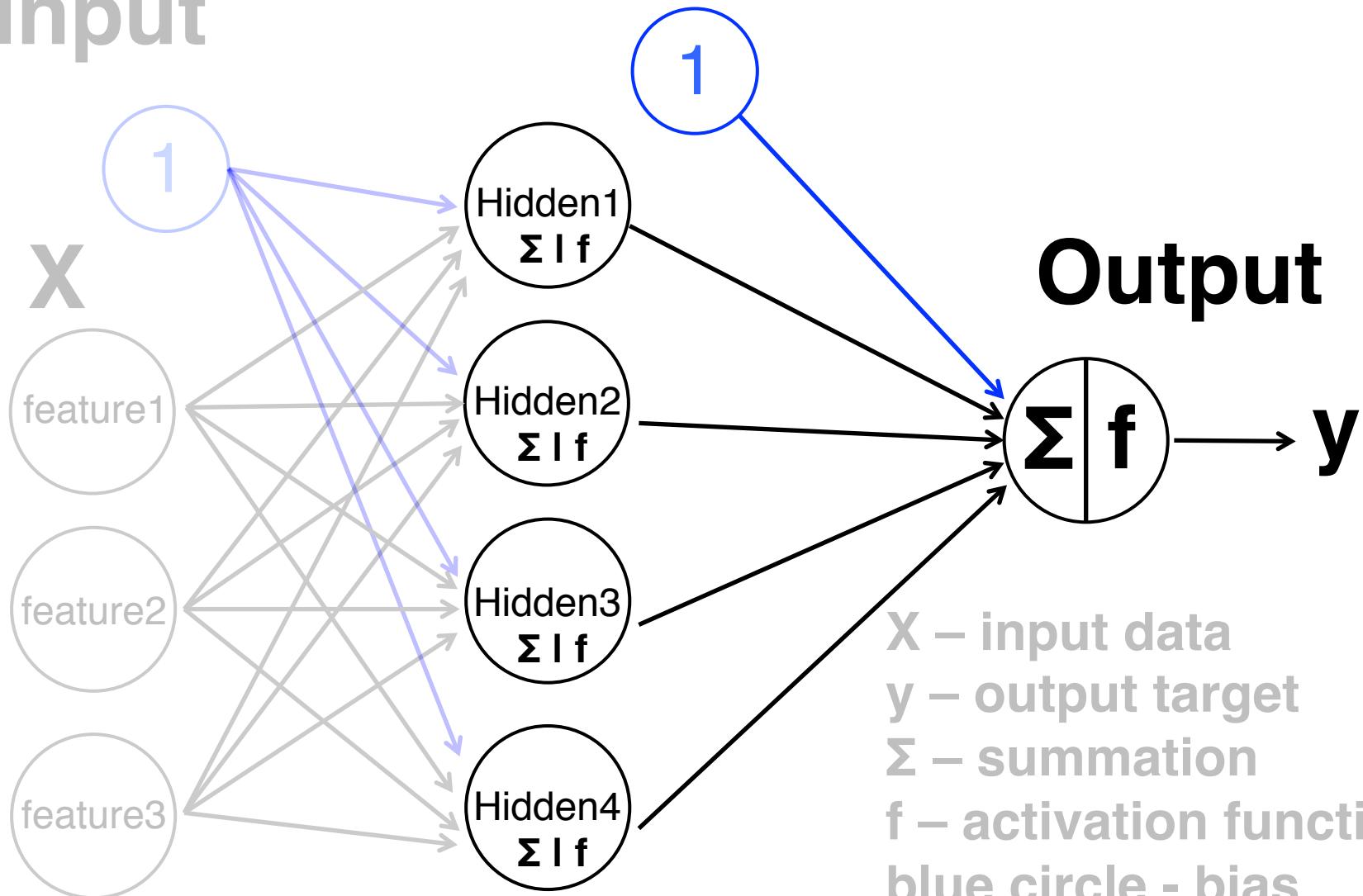
X – input data
y – output target
 Σ – summation
f – activation function
blue circle - bias

Input



X – input data
y – output target
 Σ – summation
f – activation function
blue circle - bias

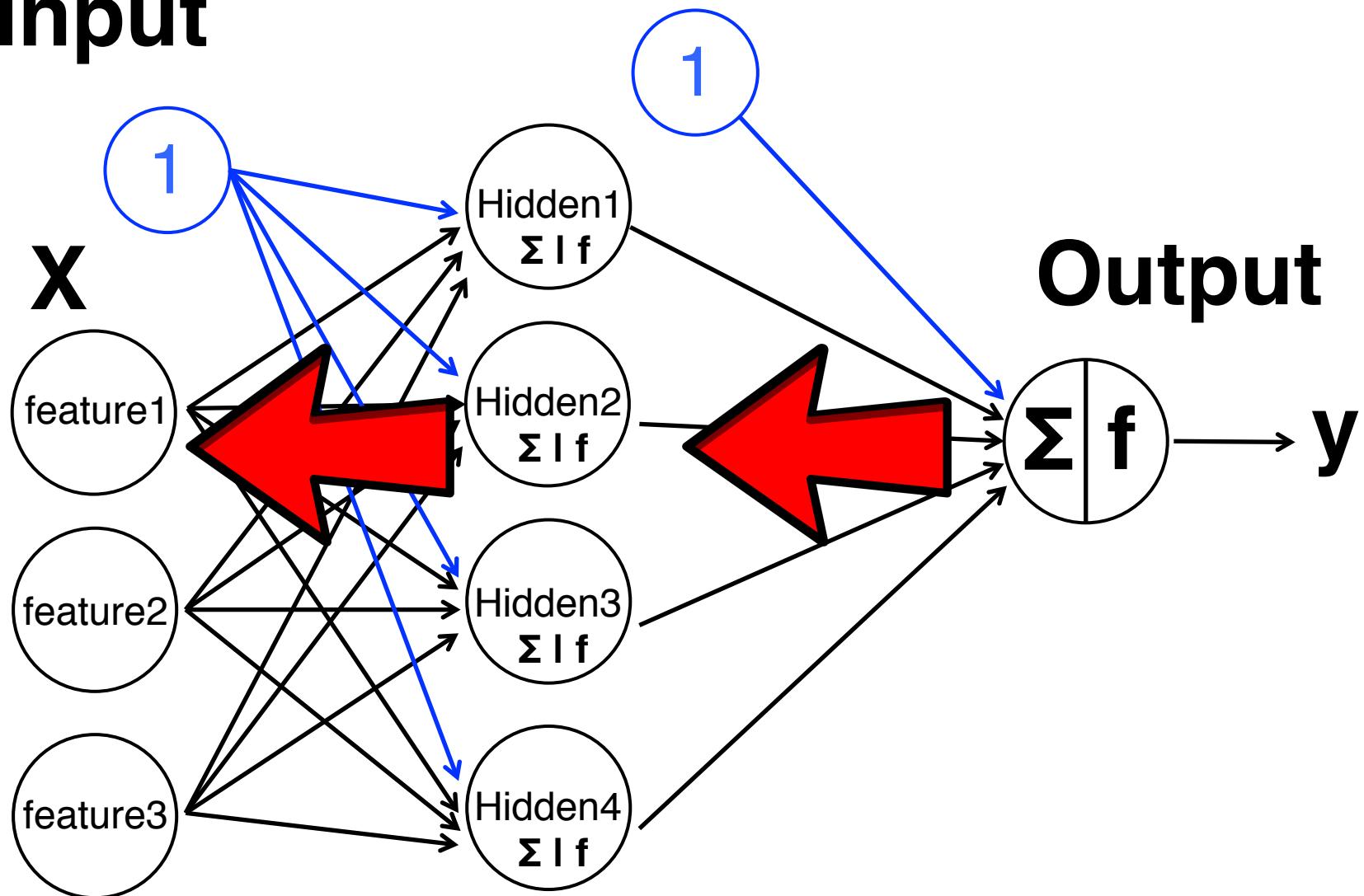
Input



Output

X – input data
y – output target
 Σ – summation
f – activation function
blue circle - bias

Input



Output

Go to notebook 02

Learning curve

Workshop time

Gentle introduction

- What's ML
- ANN history
- ANN overview

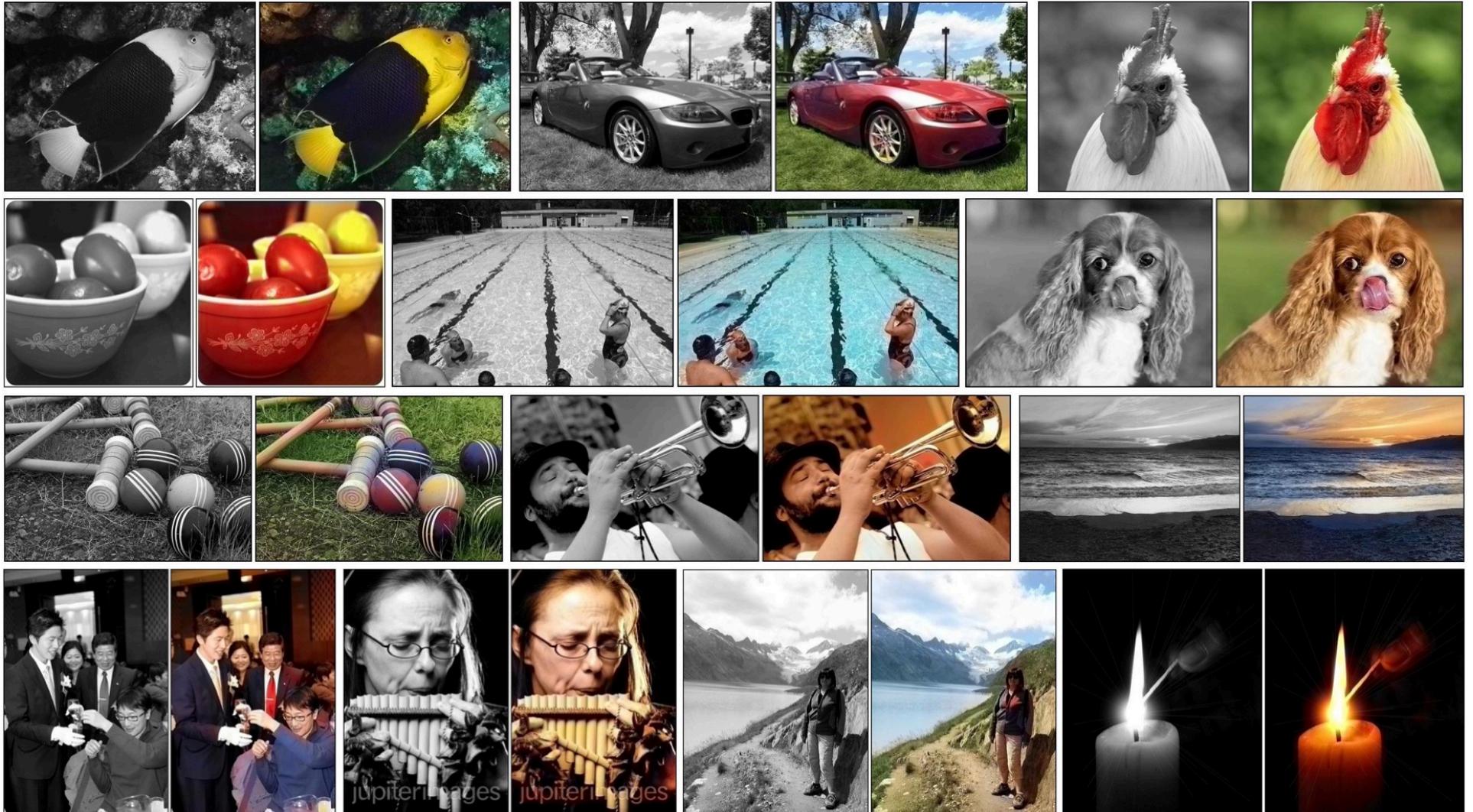
Step by step ANN

- Perceptron
- Backpropagation

Real world example

- Sklearn example

Application: Colourization



<http://whattogive.com/videoColourization/>

<http://richzhang.github.io/colorization/>

11

NEIL A. ARMSTRONG
JANET S. ARMSTRONG

2214

12/15

19 87

13.31
420

PAY TO THE
ORDER OF

Coyne's Valley Christian Academy

\$

50 00

DOLLARS



THE FIFTH THIRD BANK

CINCINNATI, OHIO 45201

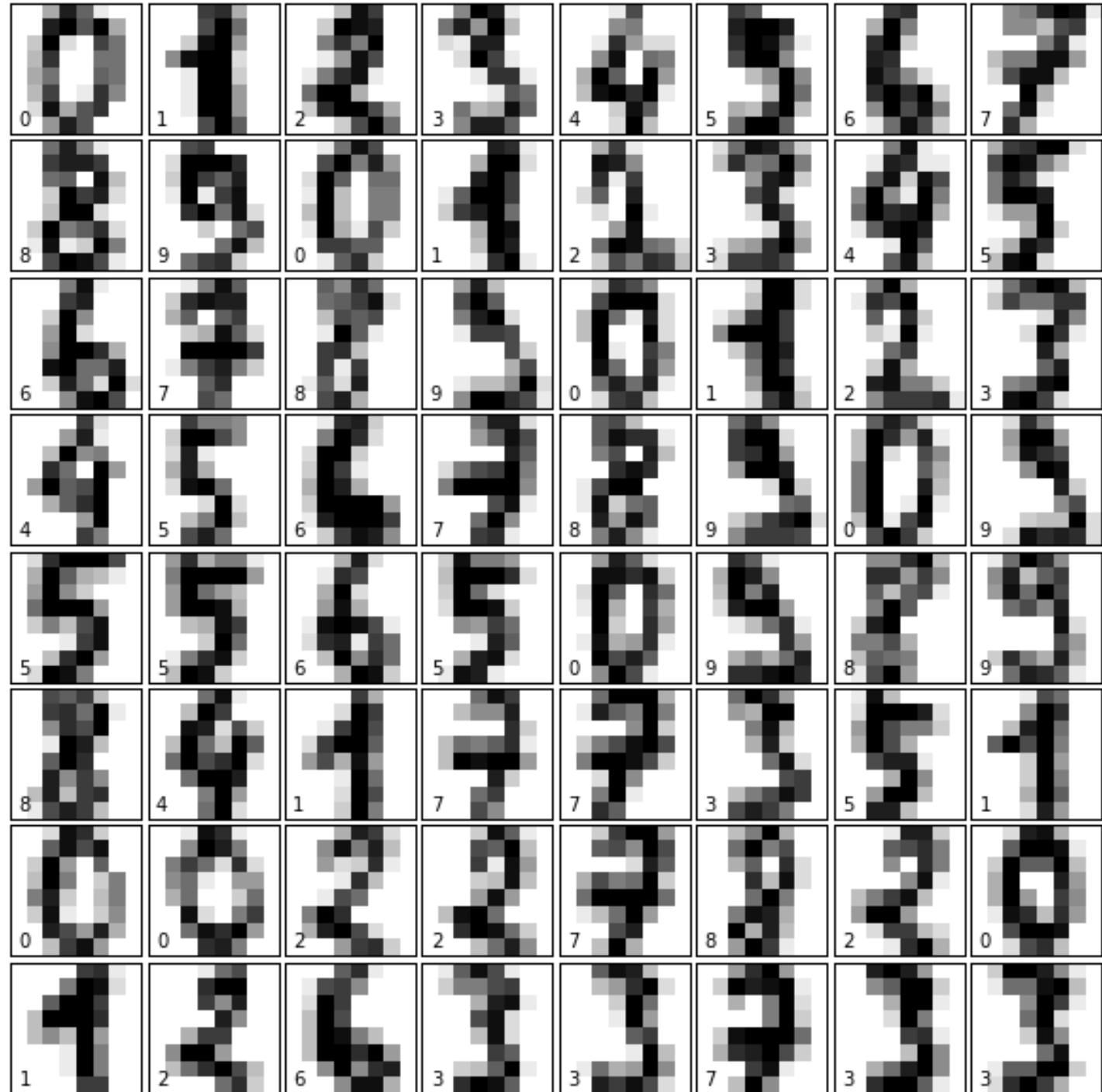
FIFTH THIRD CENTER
38 FOUNTAIN SQUARE PLAZA, CINCINNATI, OH 45263

FOR

1042000341 2214 590 536090

10000005000.00

N. A. Armstrong

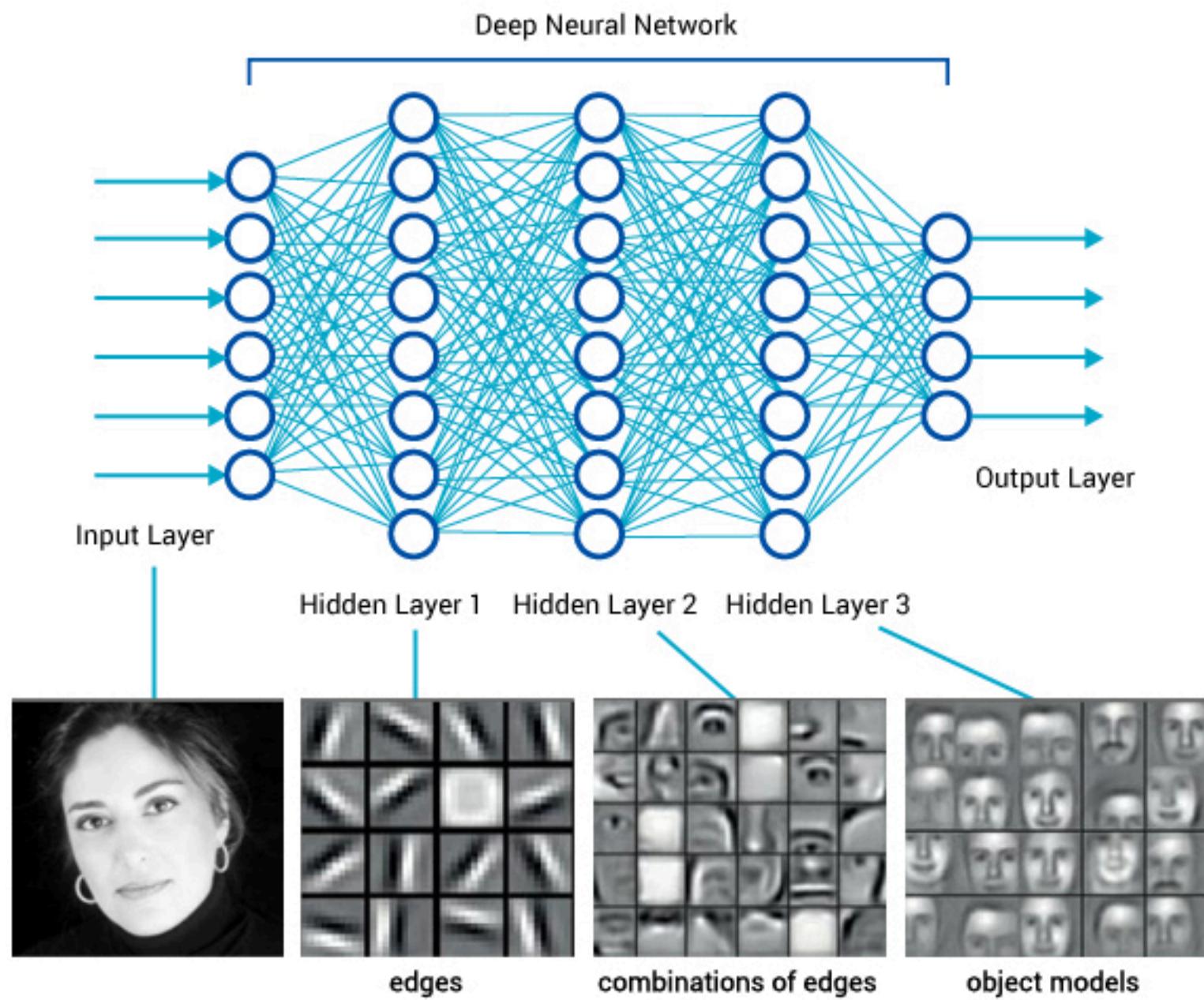


Go to notebook 03

A close-up shot from the movie Inception. Leonardo DiCaprio's character, Dom Cobb, is on the left, looking down with a serious expression. Another man's face is partially visible on the right, also looking down. The scene is dimly lit with warm, golden light.

WE NEED TO GO

DEEPER



If you want to learn more ...



<http://dlab.berkeley.edu/training>

Some useful resources

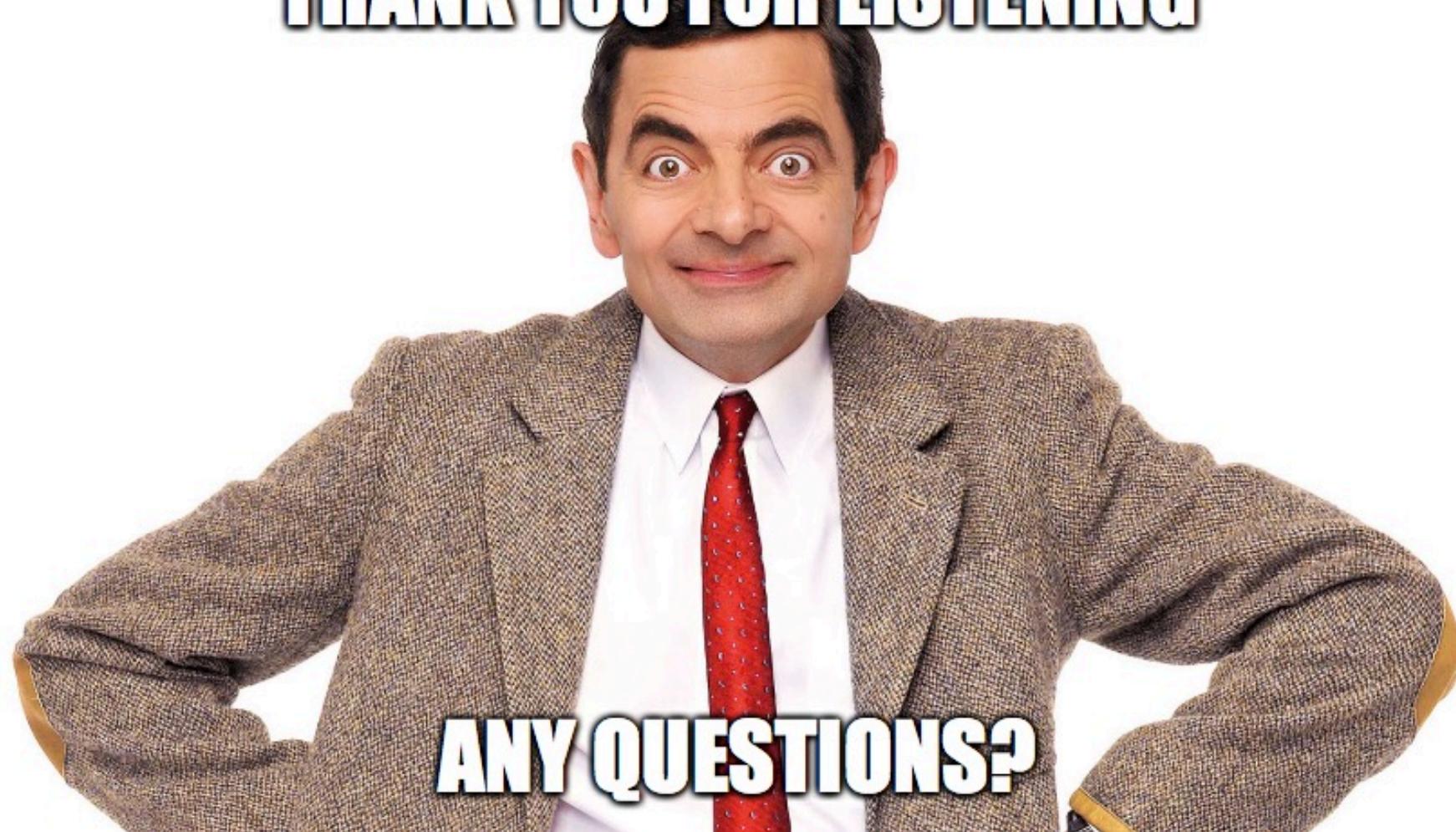
- <http://iamtrask.github.io/2015/07/12/basic-python-network/>
- <https://seat.massey.ac.nz/personal/s.r.marsland/MLBook.html>
- http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html
- <http://www.emergentmind.com/neural-network>
- <http://neuralnetworksanddeeplearning.com/>
- <https://www.coursera.org/learn/neural-networks>

You can also find most of today's workshop material on my blog:

<http://qingkaikong.blogspot.com/2016/10/machine-learning-1-what-is-machine.html>

I thank all the authors of the above links, as well as a lot of the images I got from internet.

THANK YOU FOR LISTENING



ANY QUESTIONS?