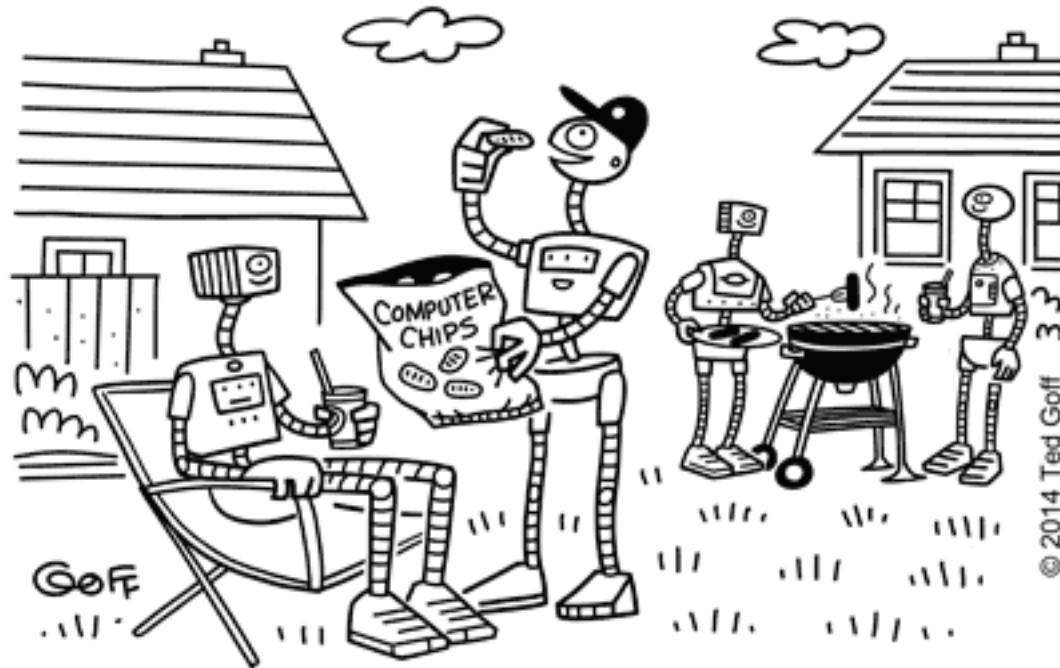


LABOR DAY BBQ 2050

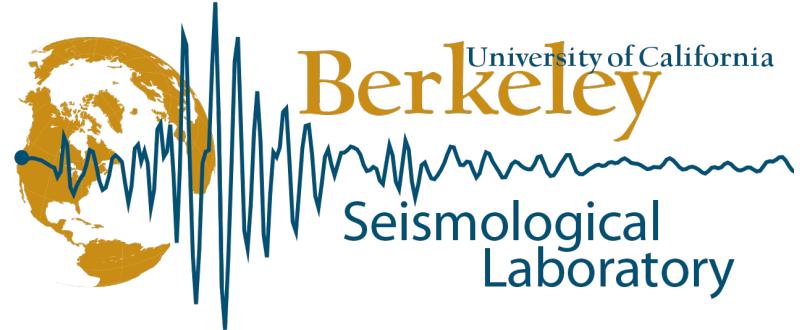


“Try one of these. They’re salty, and they come with nine new human skills.”

<https://github.com/qingkaikong/20170413 ANN basics DLab>



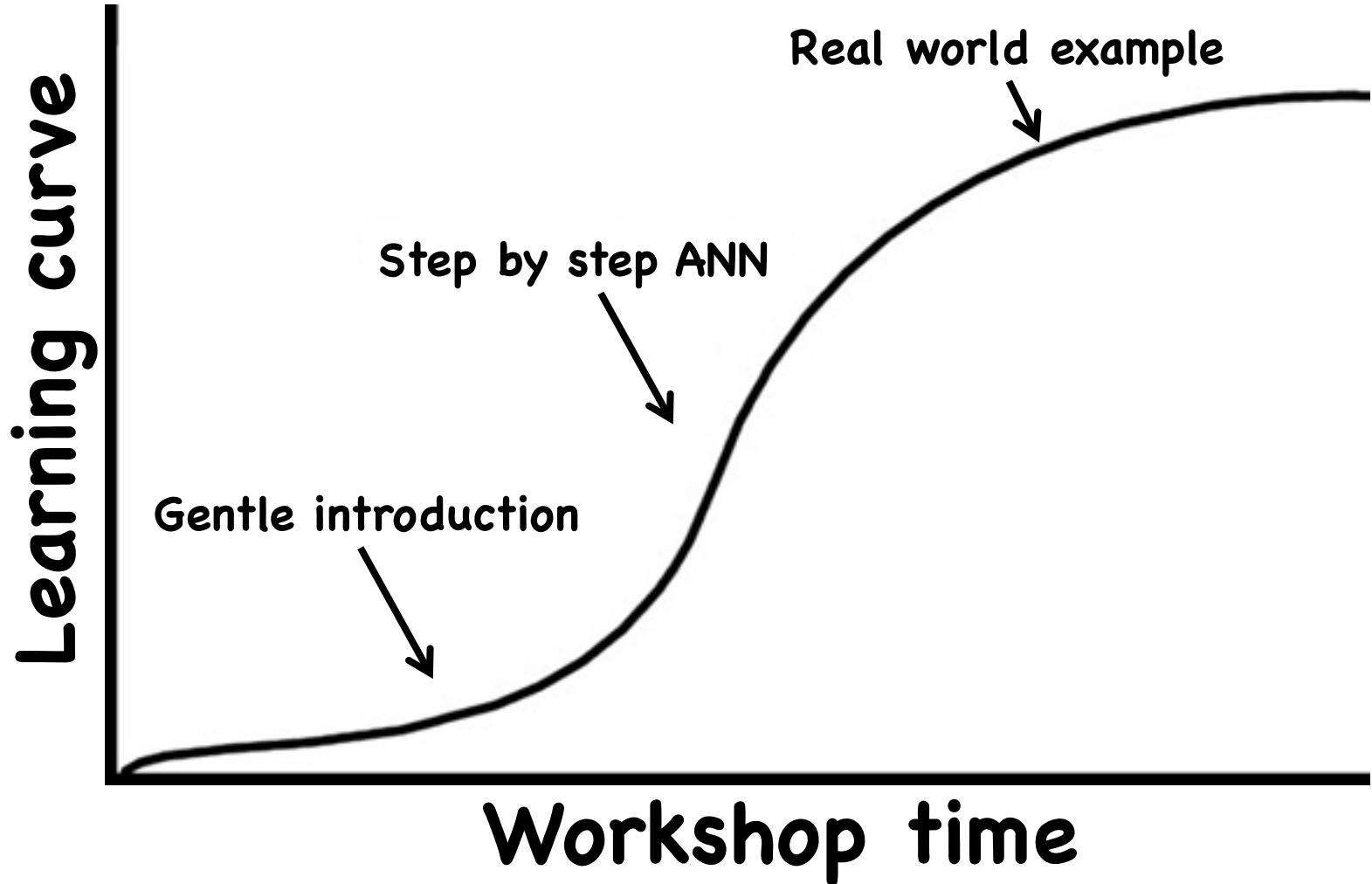
Deutsche Telekom, Inc.
Silicon Valley Innovation Center



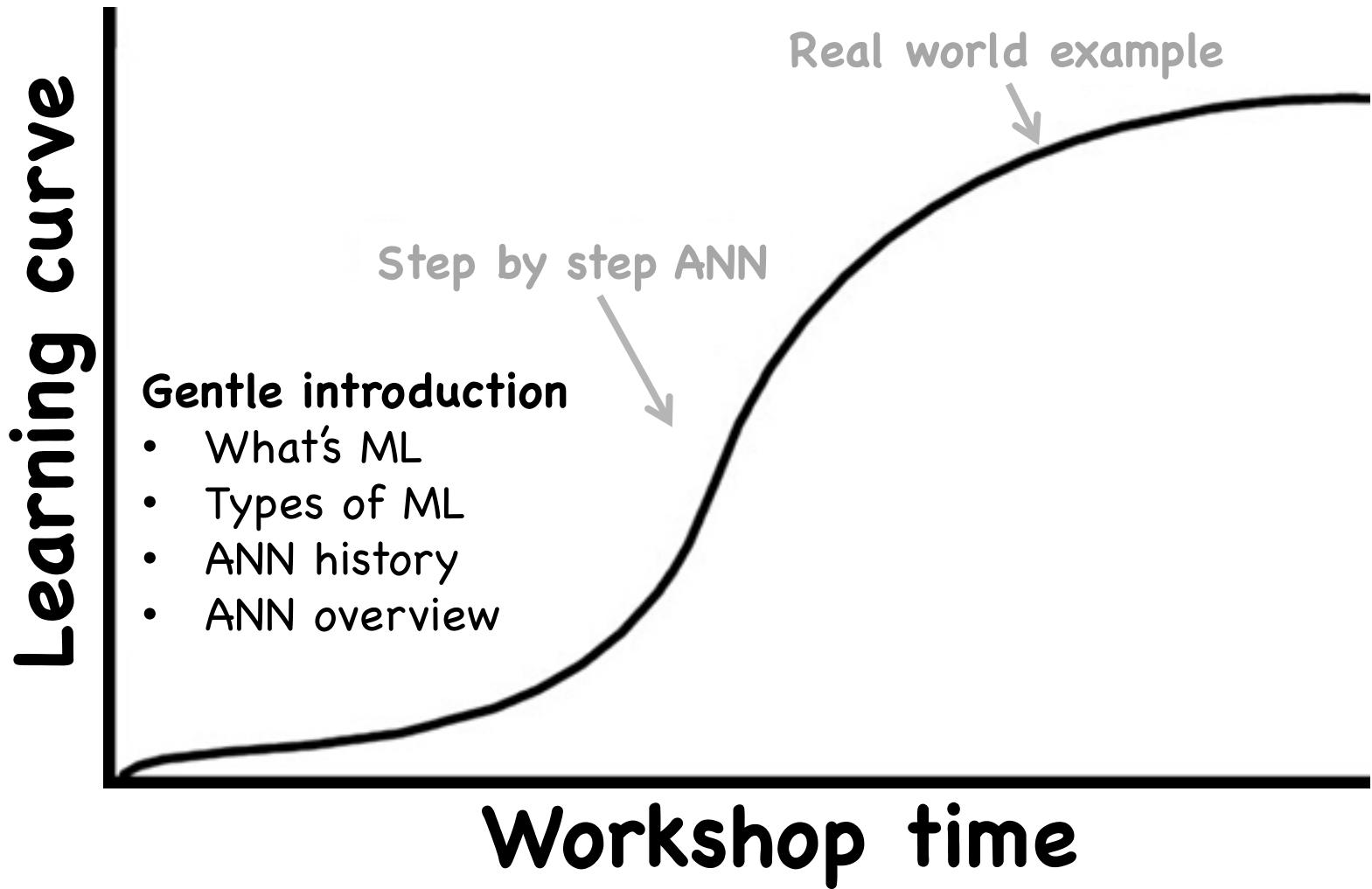
Introduction to Artificial Neural Networks

Qingkai Kong
2017-04-13

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

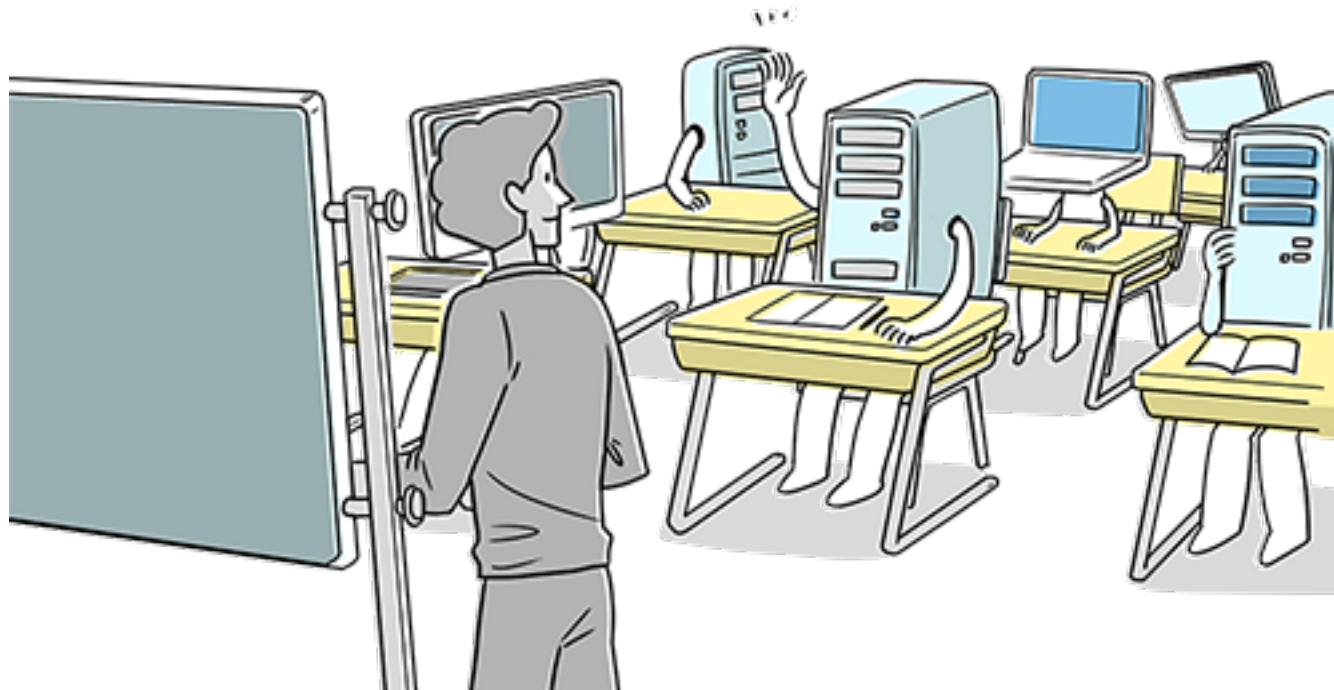


https://github.com/qingkaikong/20170413_ANN_basics_DLab



https://github.com/qingkaikong/20170413_ANN_basics_DLab

What is machine learning?



https://github.com/qingkaikong/20170413_ANN_basics_DLab



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)
---------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	-----------------------------------------------------------------------------

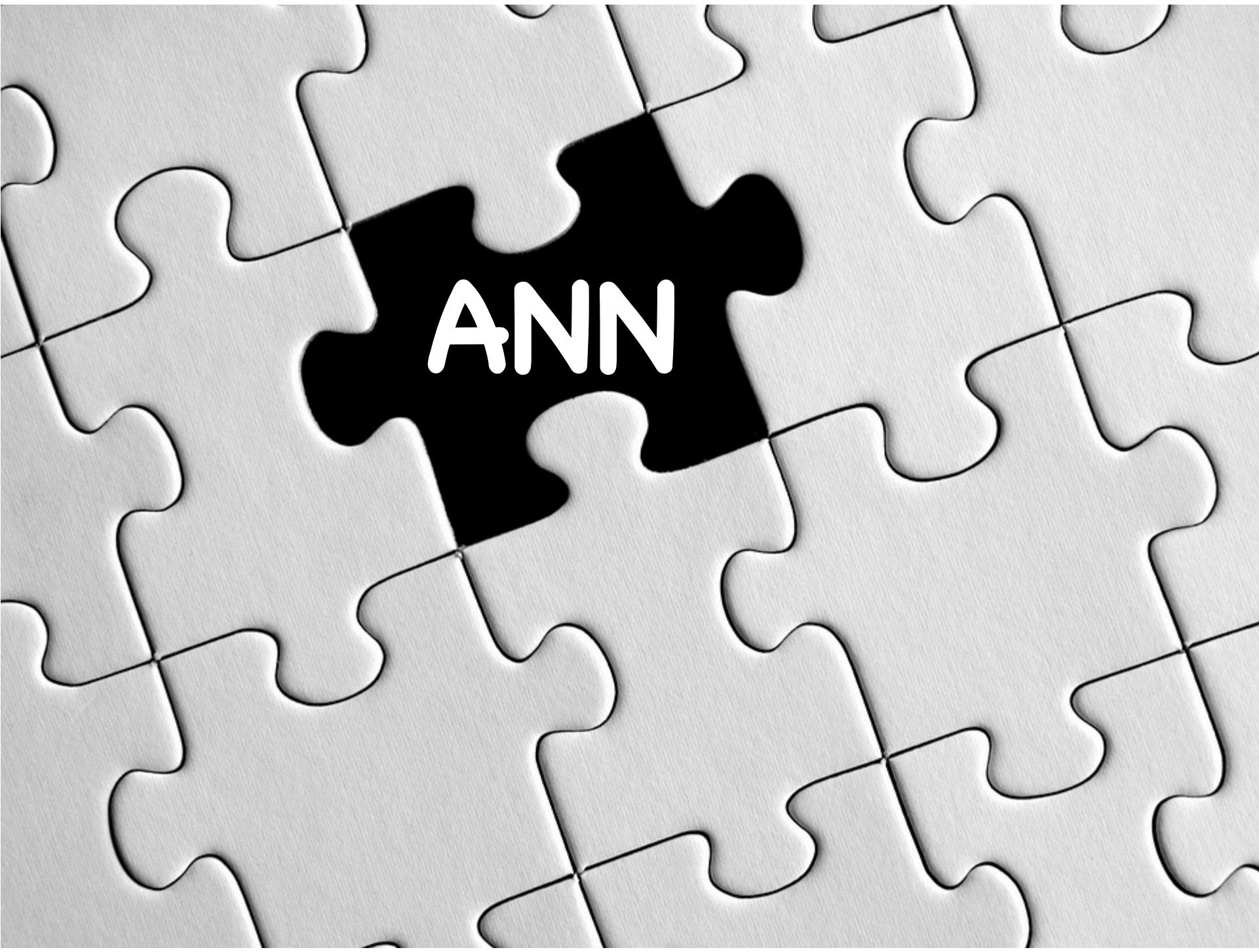
**Self-driving car
Voice recognition**

...

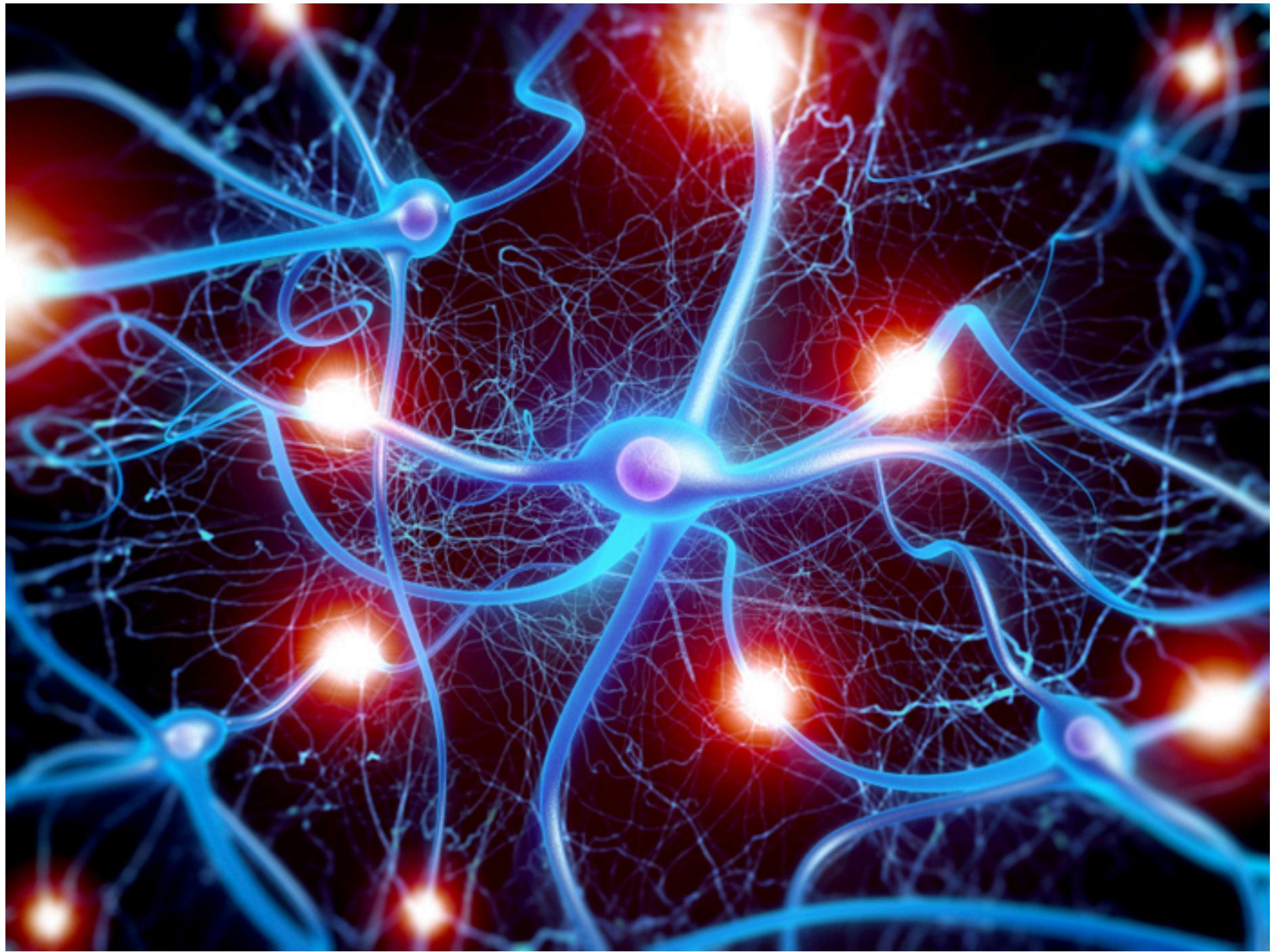
<https://github.com/qingkaikong/20170413 ANN basics DLab>

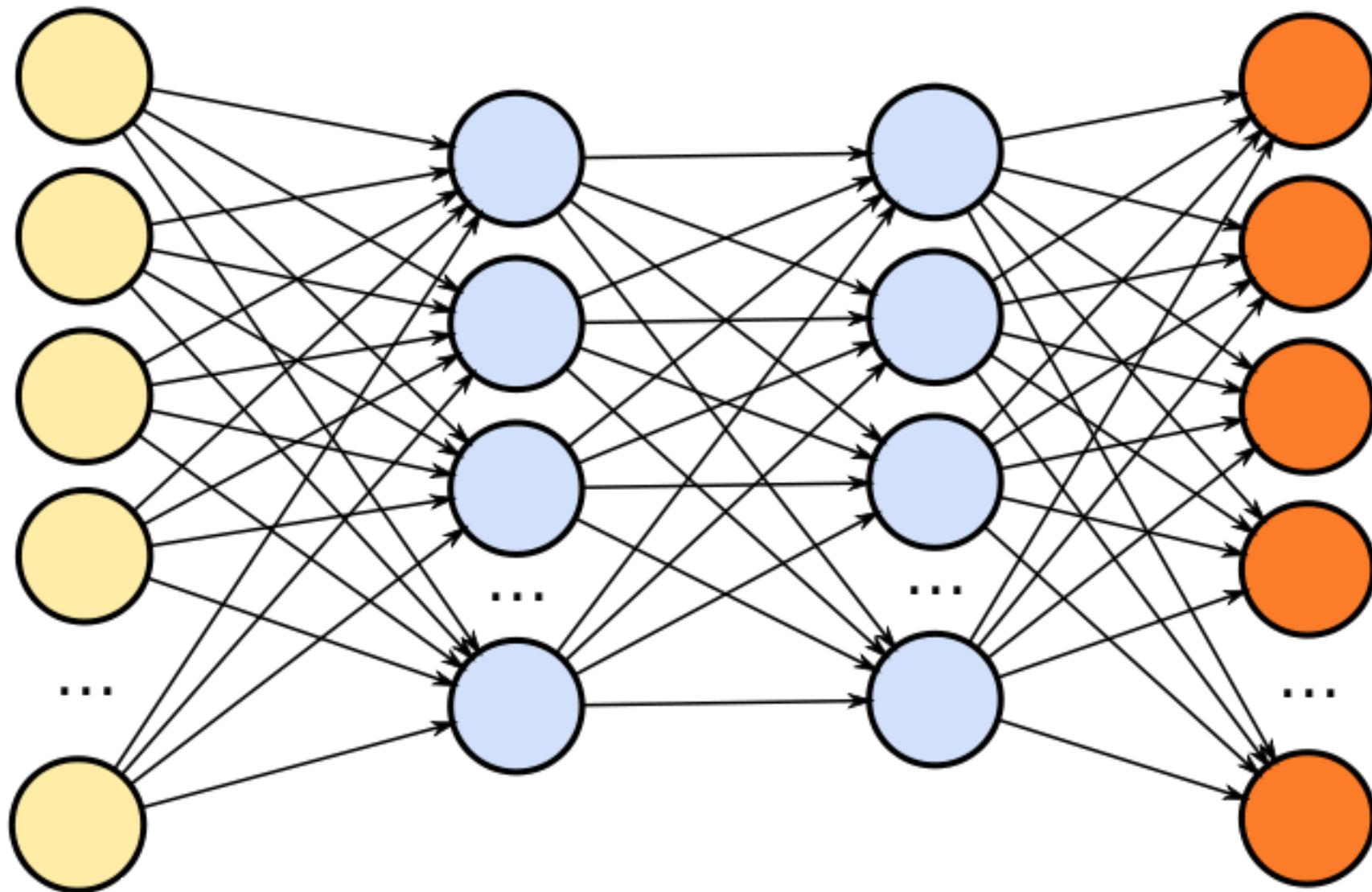
Not always
working

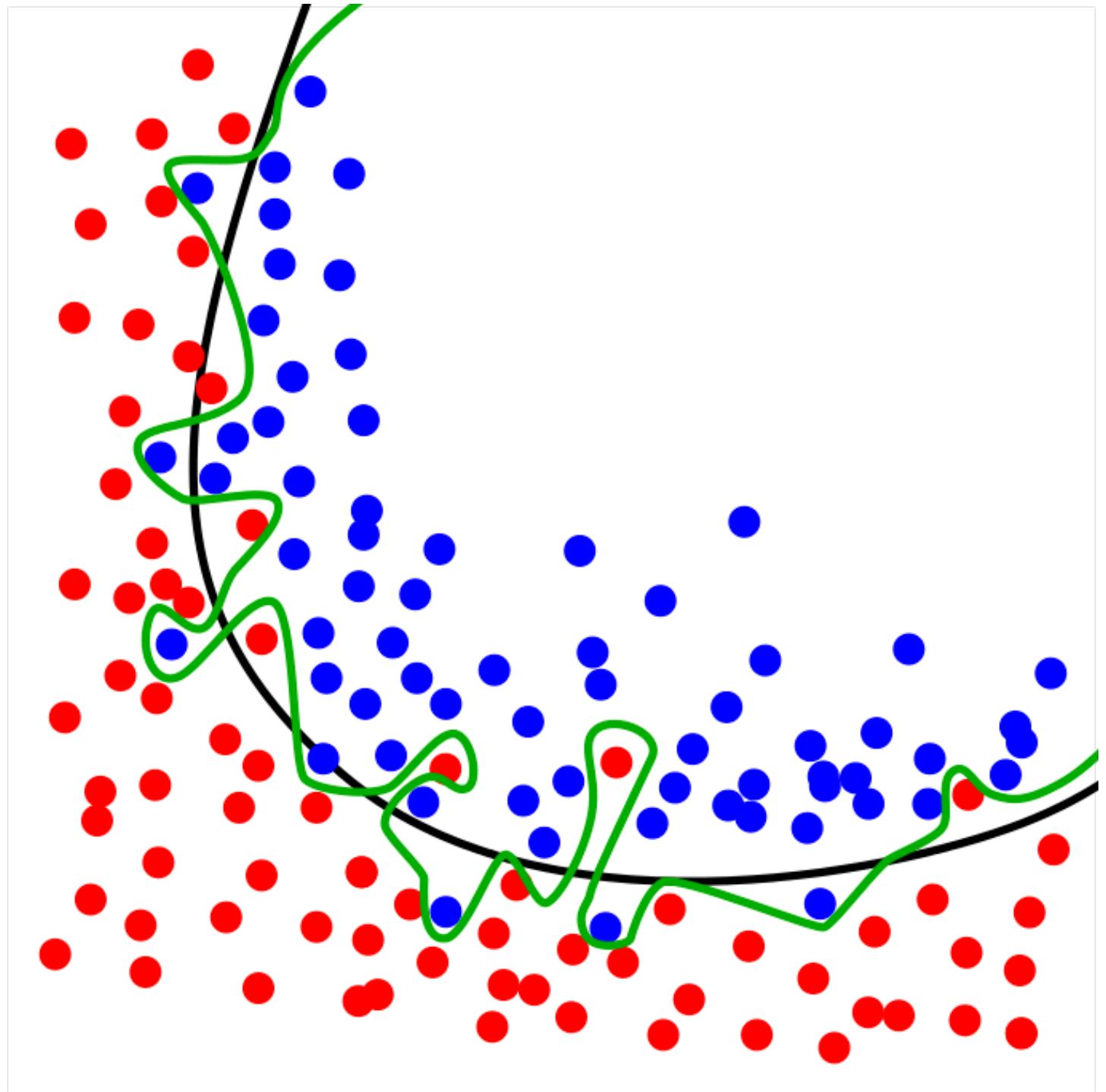




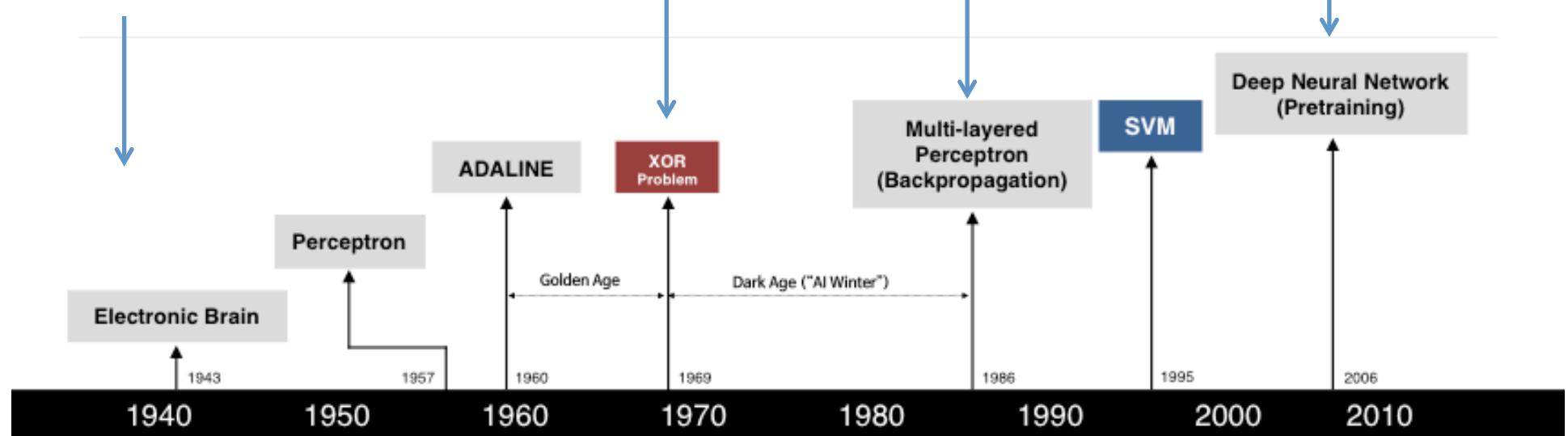
ANN







1940s Birth



S. McCulloch - W. Pitts



- Adjustable Weights
- Weights are not Learned



F. Rosenblatt



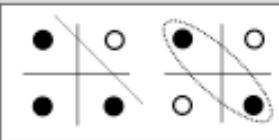
B. Widrow - M. Hoff



- Learnable Weights and Threshold



M. Minsky - S. Papert



• XOR Problem



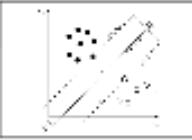
D. Rumelhart - G. Hinton - R. Williams



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



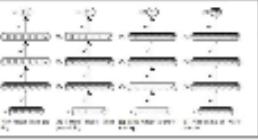
V. Vapnik - C. Cortes



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

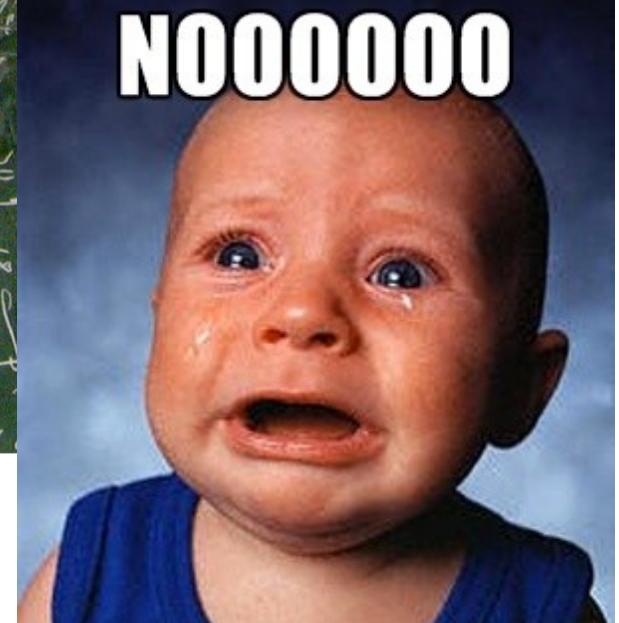


G. Hinton - S. Ruslan



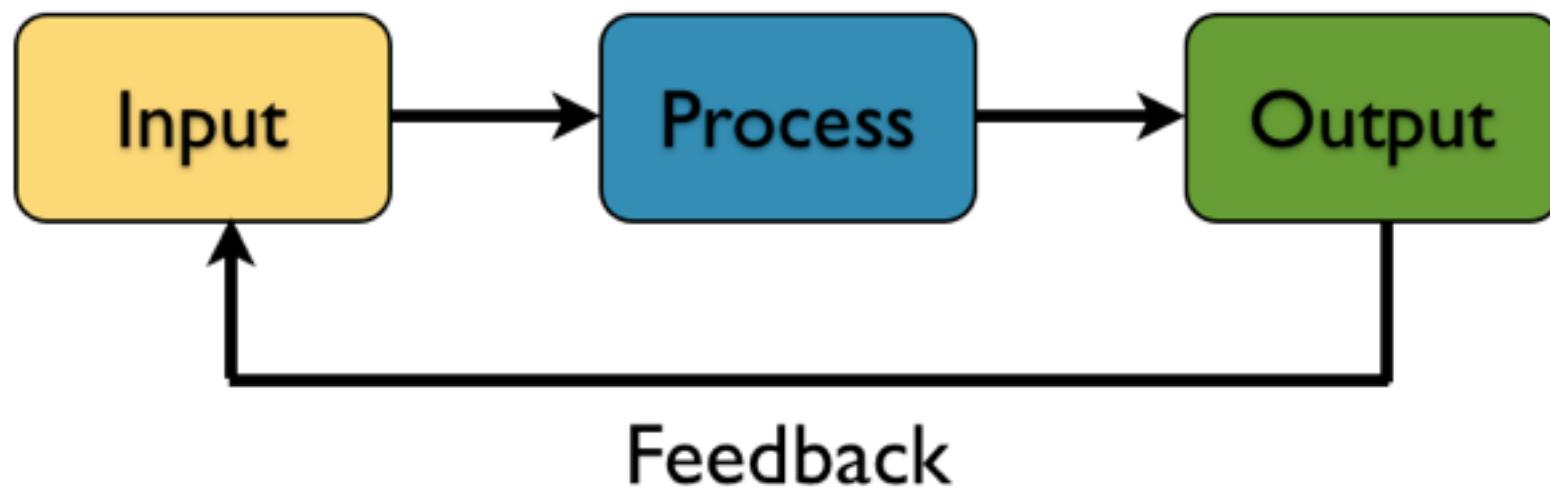
• Hierarchical feature Learning

The image shows a blackboard covered in mathematical notation, including symbols like ψ , f , $\mathcal{L}_E(\mathcal{P})$, and various algebraic and geometric diagrams. In the bottom right corner, there is a large, distressed photograph of a baby's face with the word "NOOOOO" written in large, bold, white letters across it.

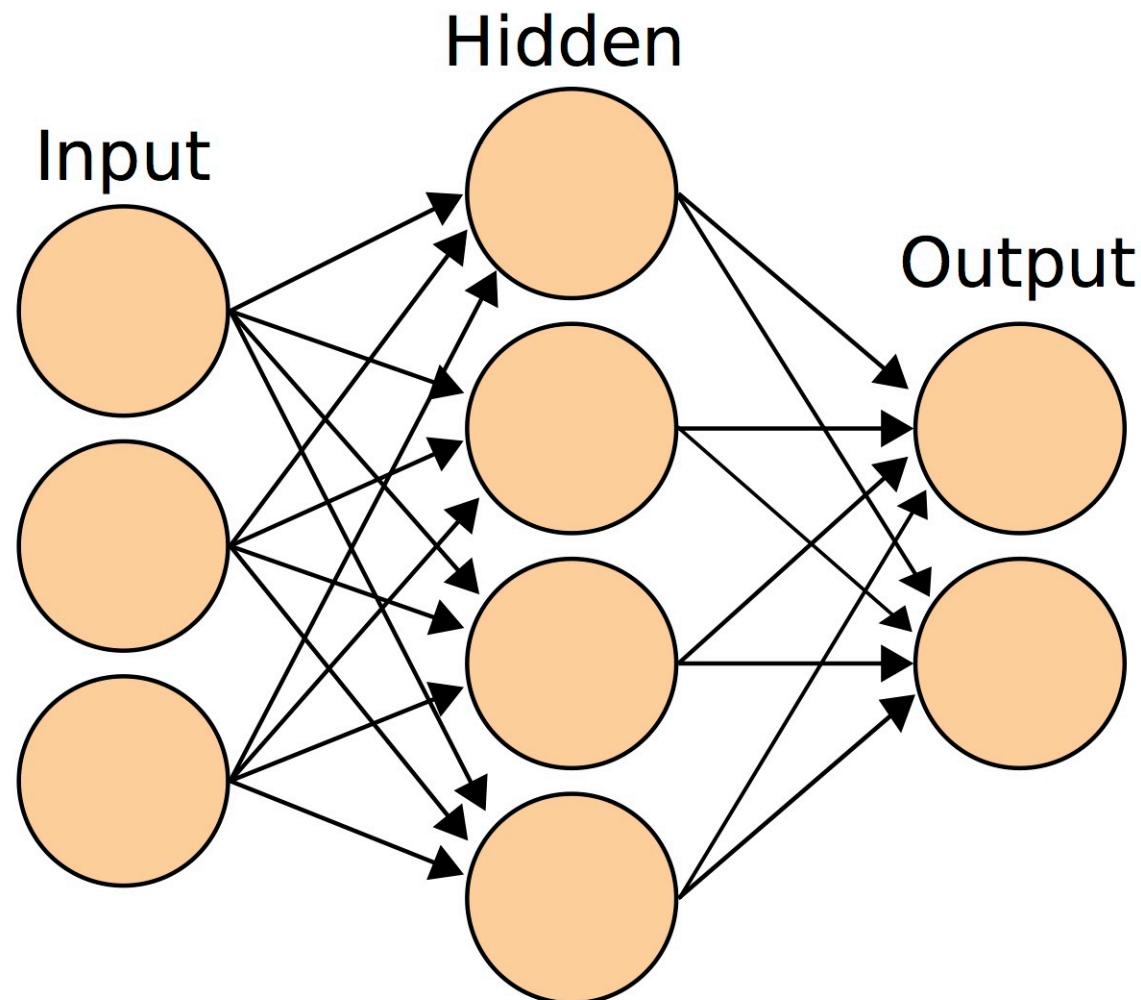




ANN in simple view



ANN jargons



What're the weights





YOU'RE IN MY SPOT

GRAPHICS GARAGE

Input



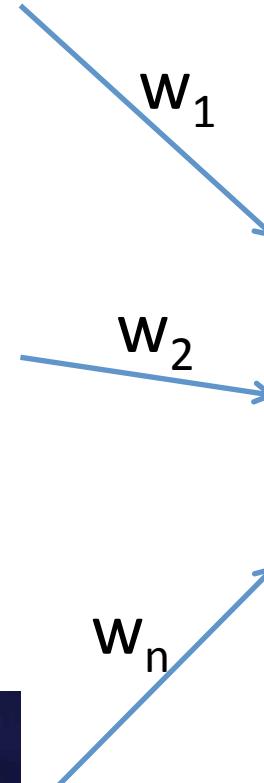
.

.

.



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

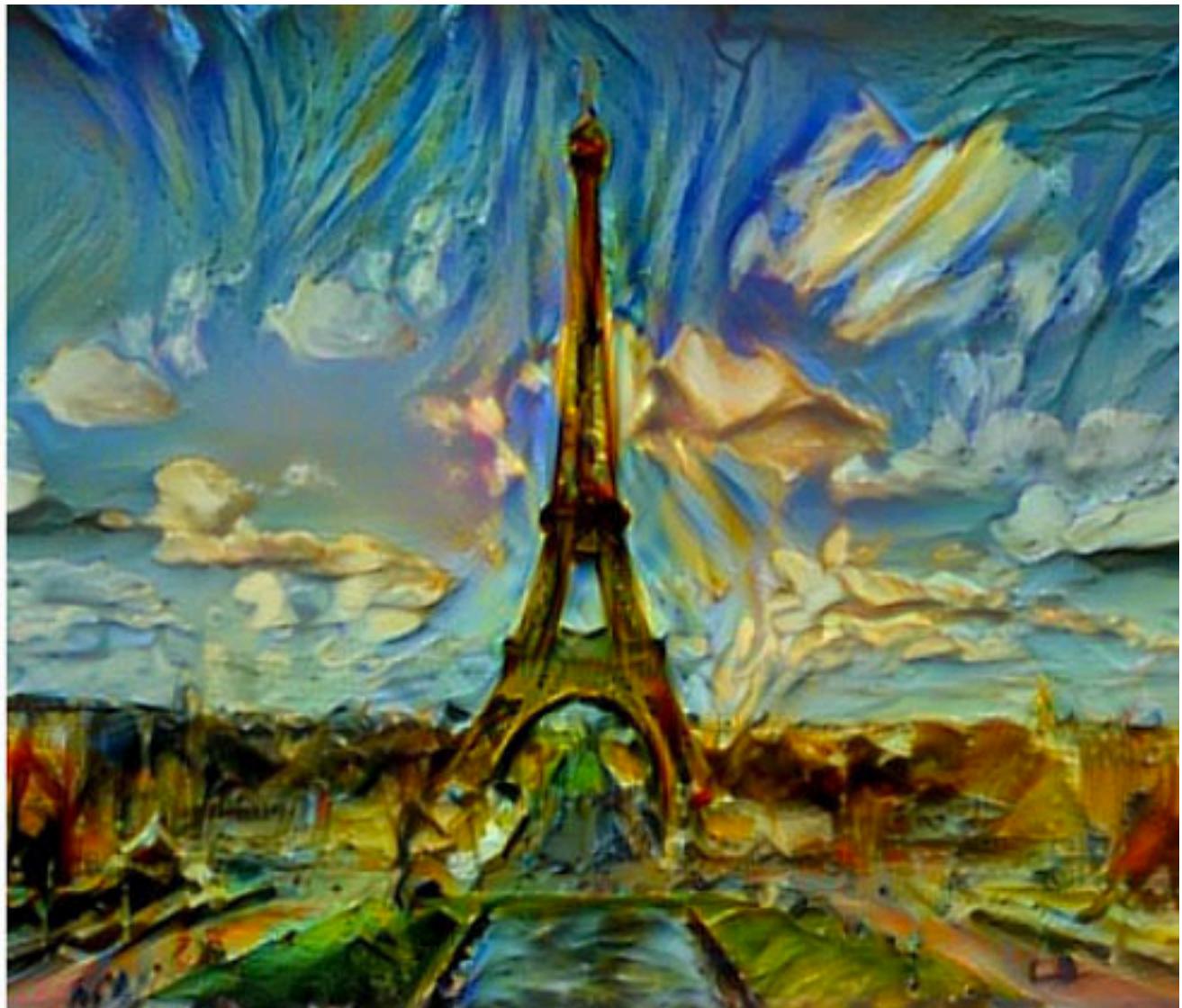
Step by step ANN

- Perceptron
- Backpropagation

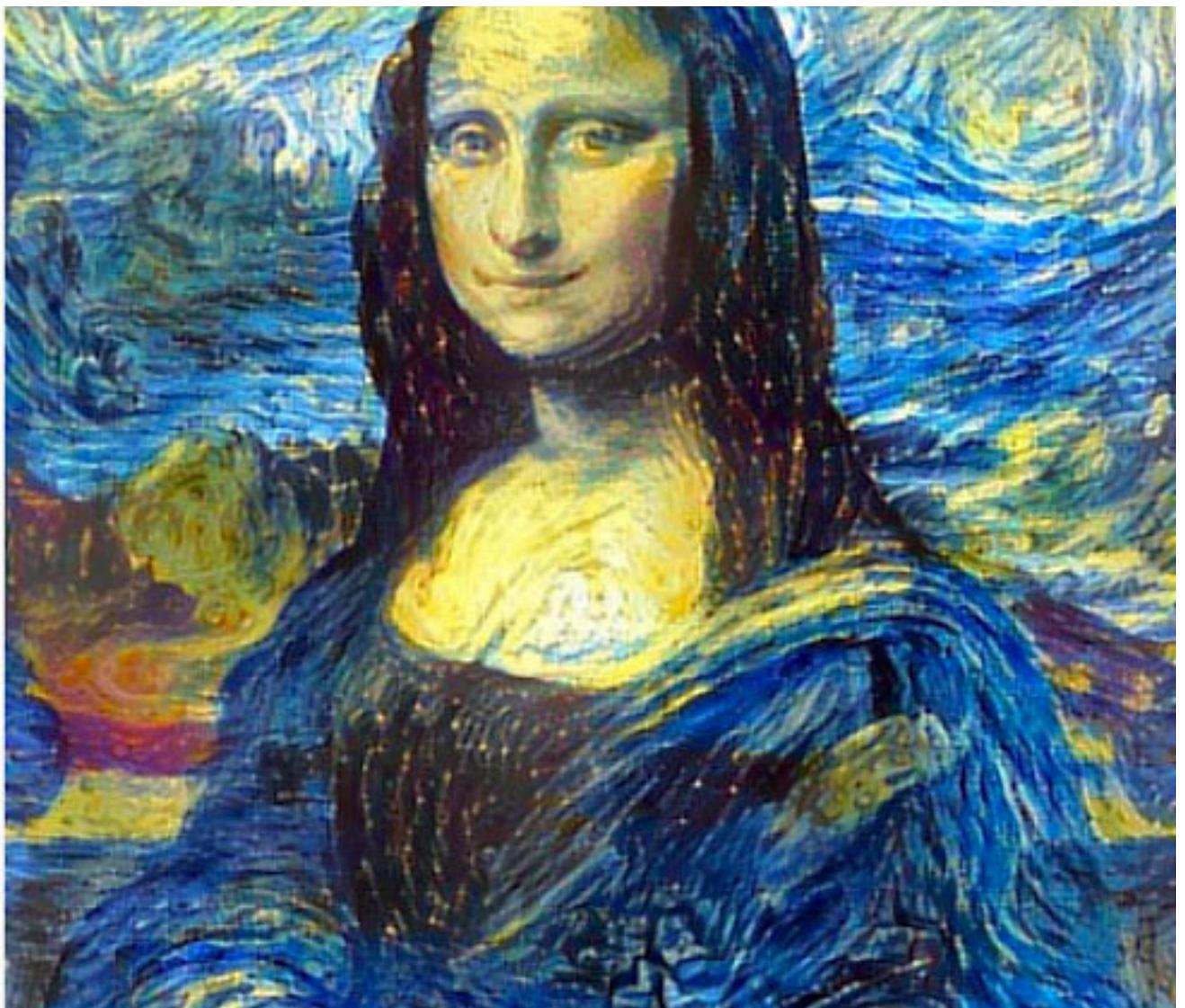
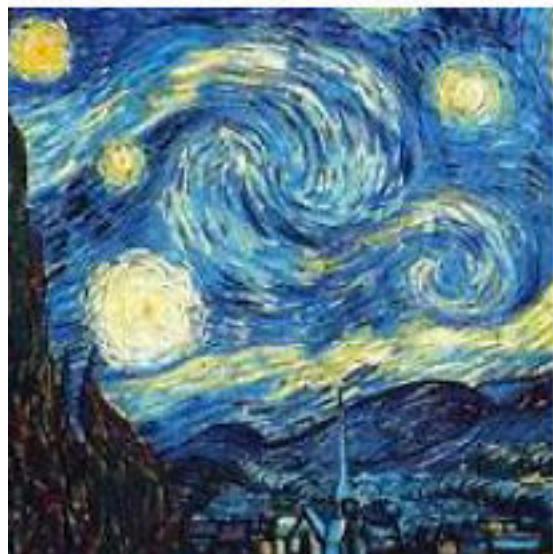
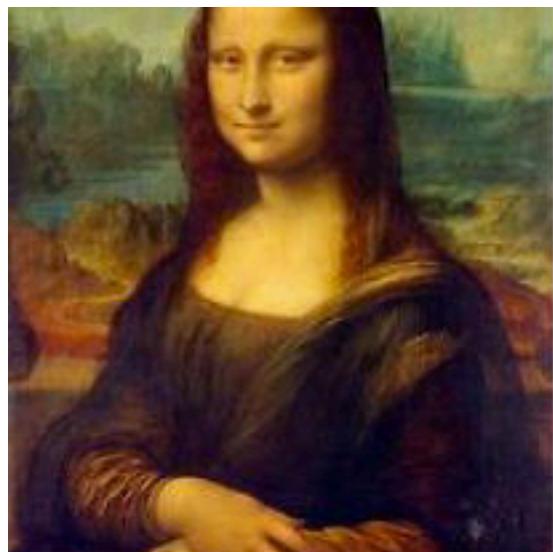
Real world example



Application: Learn arts

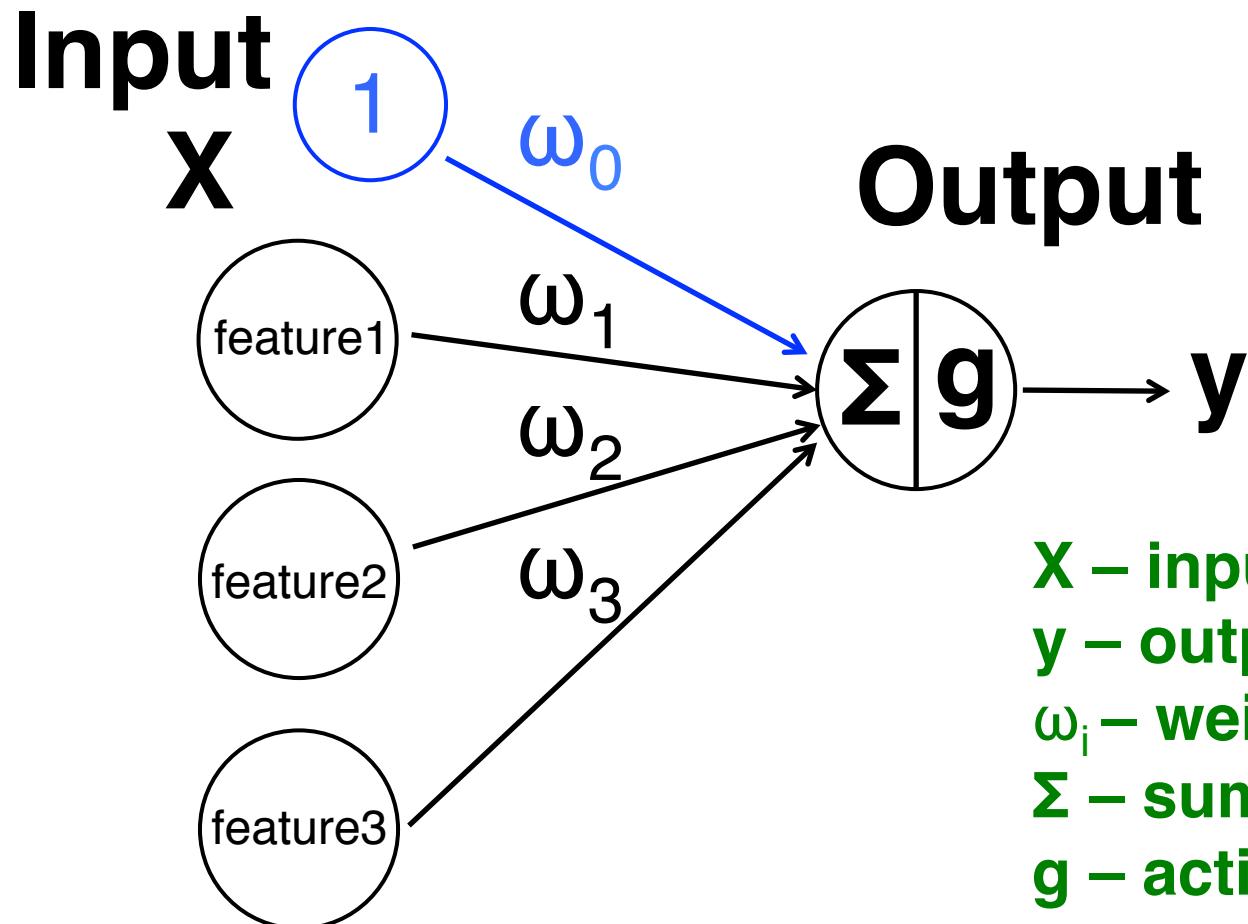


<https://arxiv.org/abs/1508.06576v1>
<https://deepoch.io/>



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



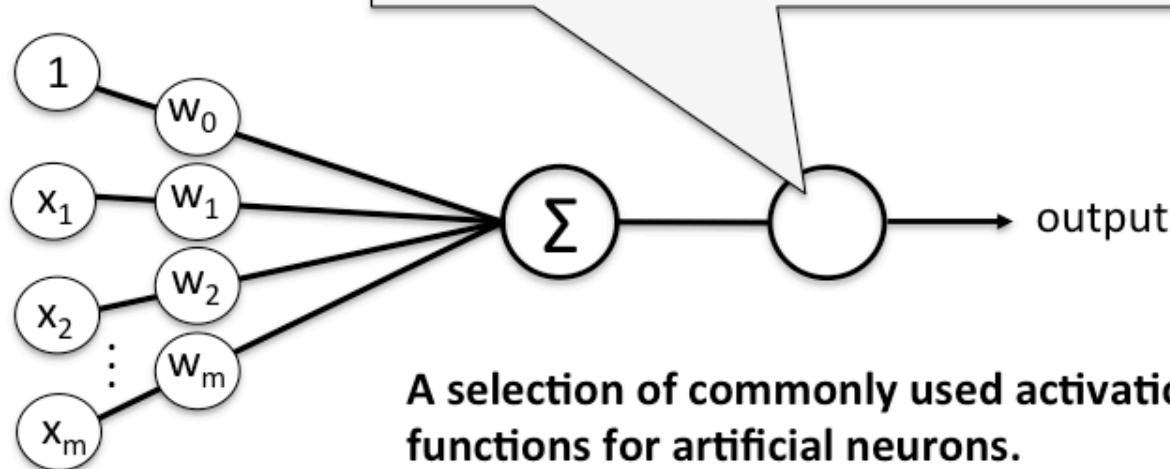


X – input data
y – output target
 ω_i – weights
 Σ – summation
g – activation function
Blue circle – bias

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

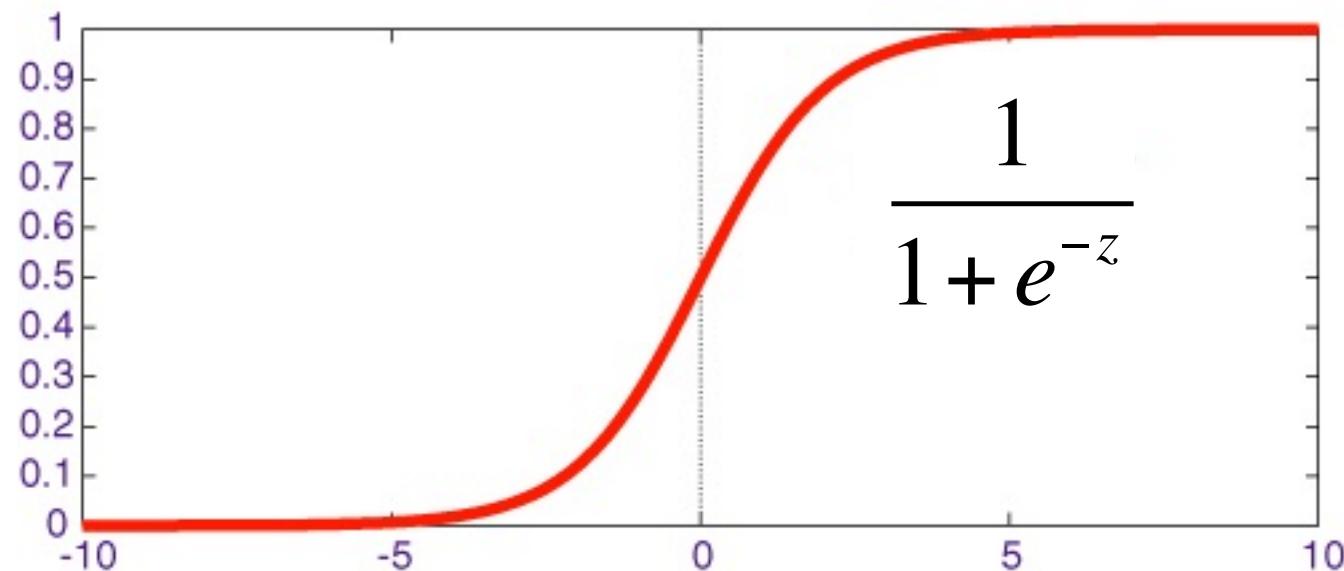
$$y = g(\omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n)$$

	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 on activation function

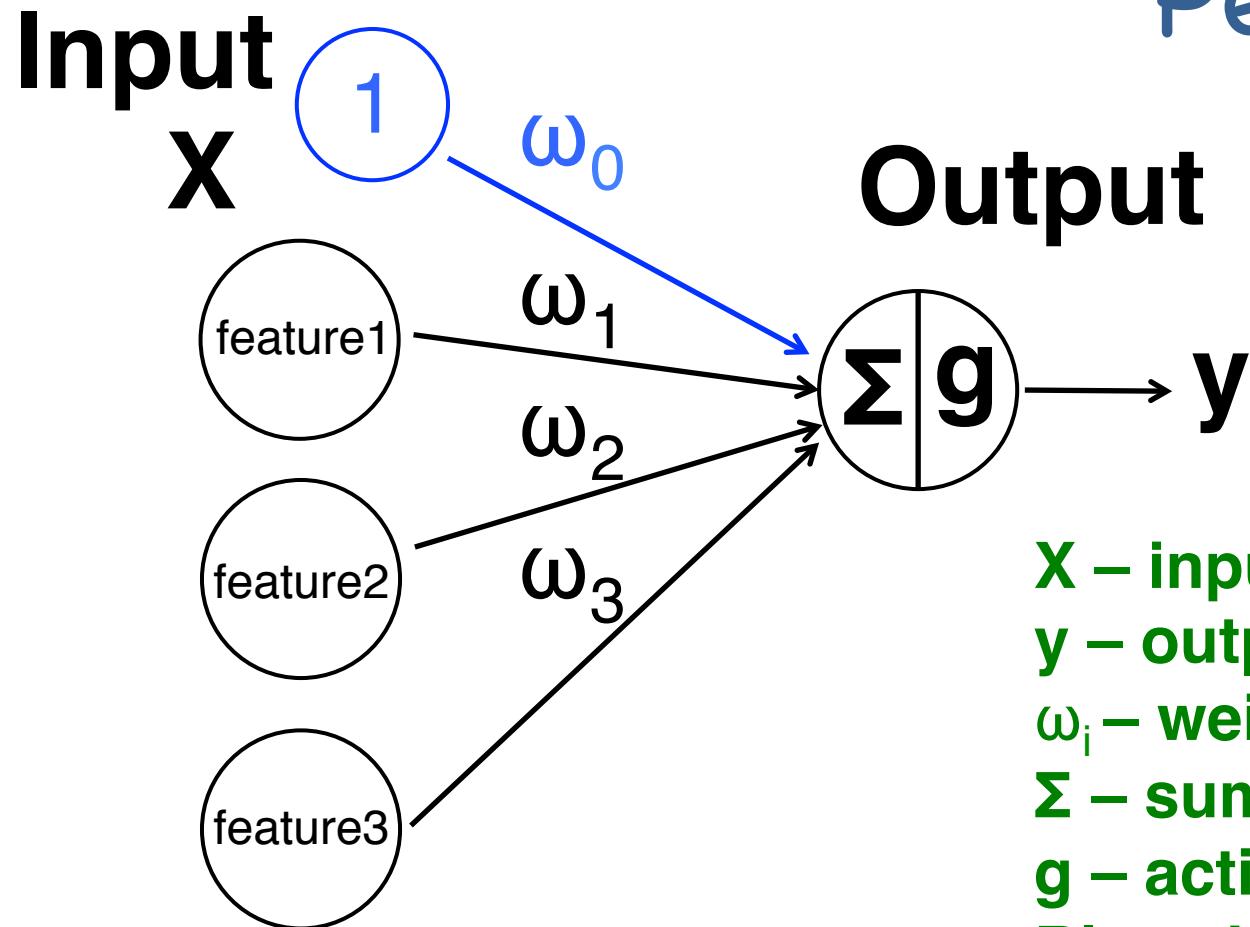


$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n$$

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

$$\frac{dg(z)}{dx} = g(z)(1 - g(z))$$

Perceptron



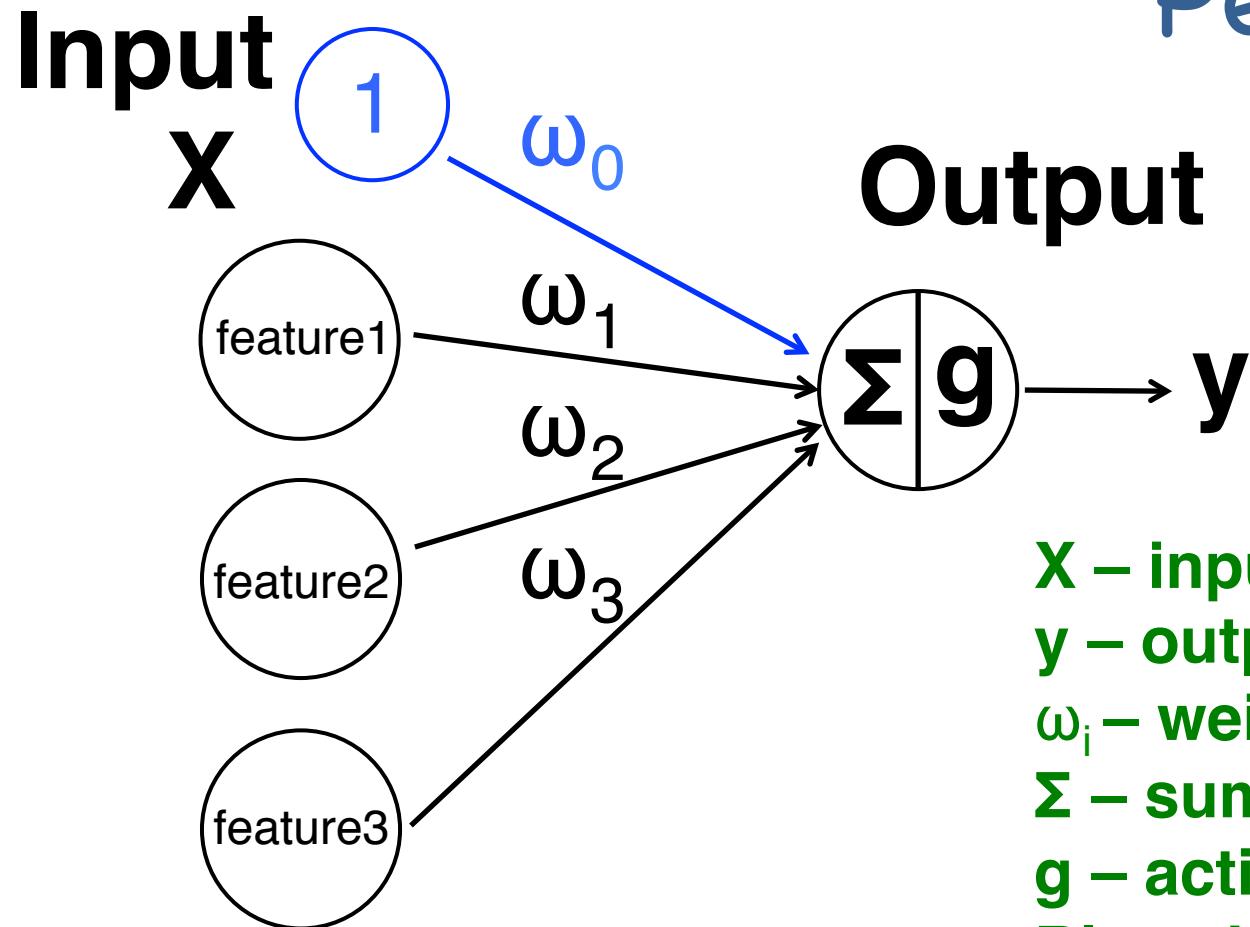
X – input data
y – output target
 ω_i – weights
 Σ – summation
g – activation function
Blue circle – bias

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

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

y This is our estimation

Perceptron



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

Error = Target - Estimation

~~Error~~



How the ANN learns

Error = Target - Estimation

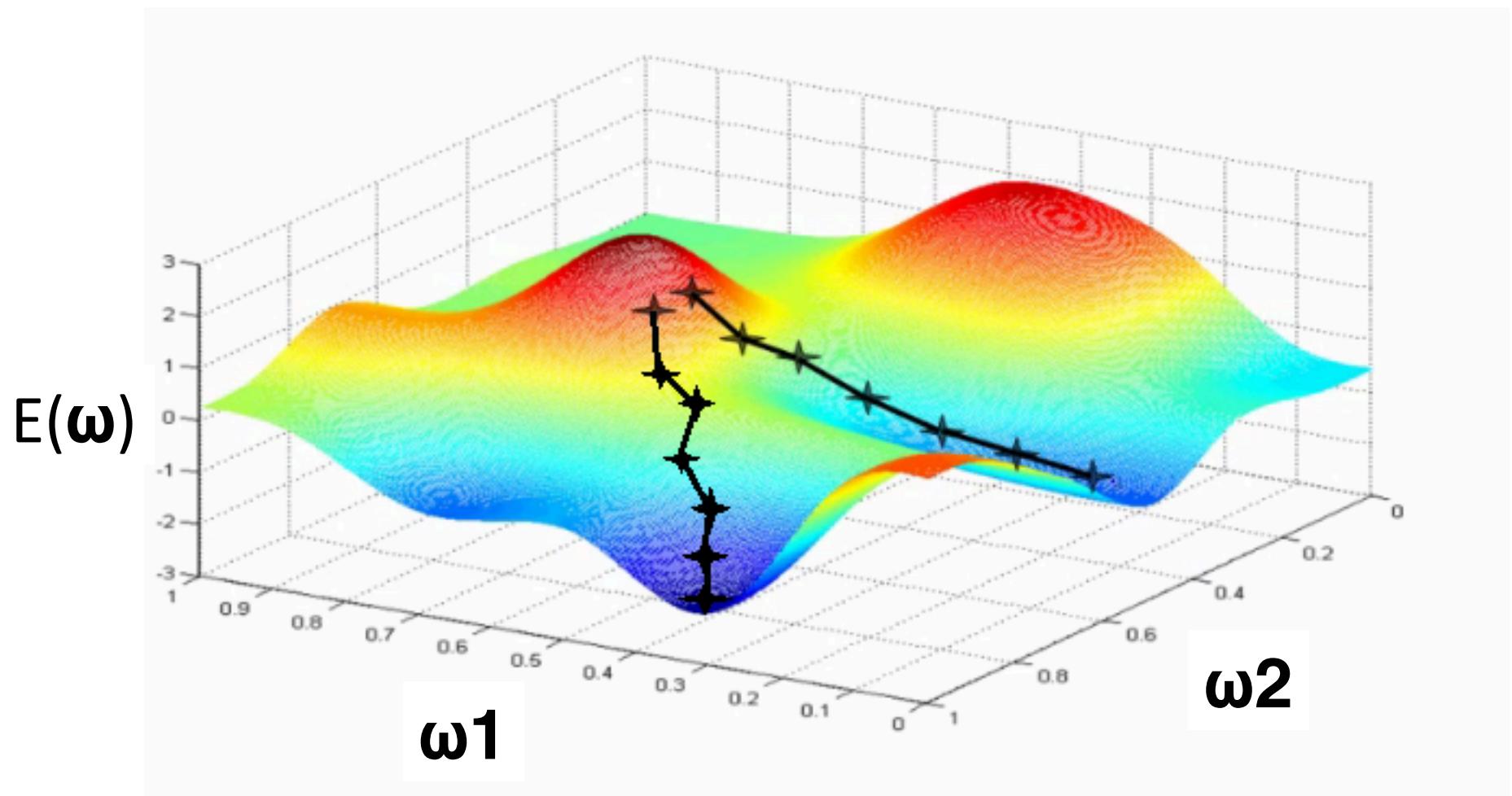
Learning:

- Measure error
- Update weights to reduce error next time!



HOW?





View from Gradient Descent

$$Error = \sum_{k=1}^N (y_k - t_k)$$

y_k – estimation
 t_k – target
 Σ – summation

Estimation 0, Target 1, Error = -1

Estimation 1, Target 0, Error = 0

If we add them, we got error 0

View from Gradient Descent

Sum-of-Squares error

$$E(w) = \frac{1}{2} \sum_{k=1}^N (y_k - t_k)^2 = \frac{1}{2} \sum_{k=1}^N [g(\sum_{i=0}^L \omega_{ik} x_i) - t_k]^2$$

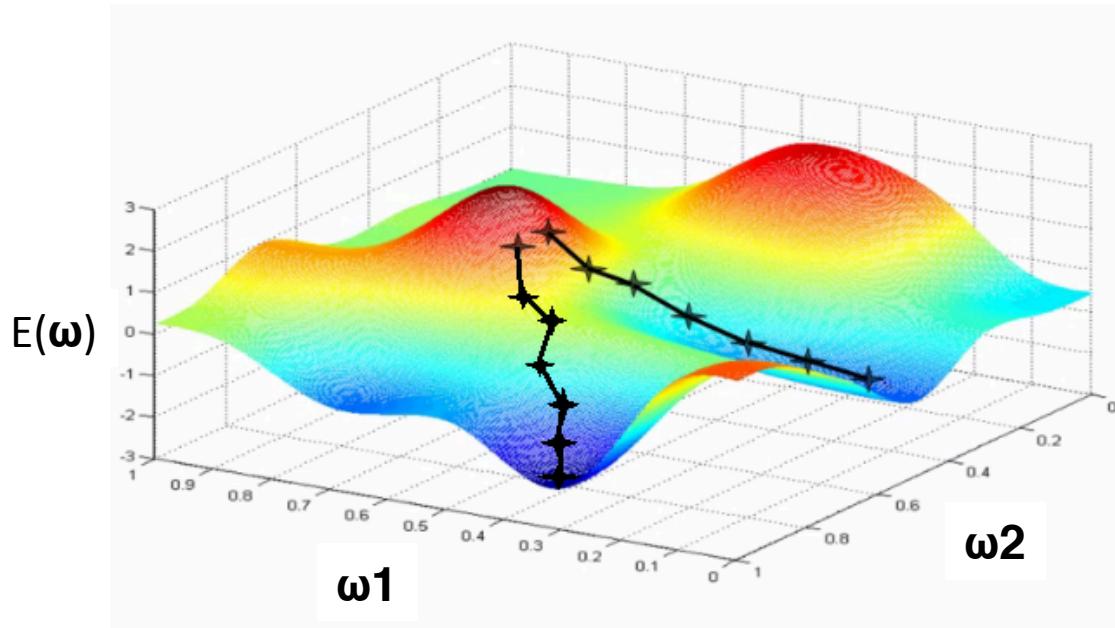
Where $y_k = g(\sum_{i=0}^L \omega_{ik} x_i)$

y_k – estimation
t_k – target
Σ – summation
g – activation function
x_i – input
ω - weights

View from Gradient Descent

Sum-of-Squares error

$$E(w) = \frac{1}{2} \sum_{k=1}^N (y_k - t_k)^2 = \frac{1}{2} \sum_{k=1}^N [g(\sum_{i=0}^L \omega_{ik} x_i) - t_k]^2$$



y_k – estimation
t_k – target
Σ – summation
g – activation function
x_i – input
ω - weights

View from Gradient Descent

$$\begin{aligned}\frac{\partial E}{\partial \omega_{ik}} &= \frac{\partial}{\partial \omega_{ik}} \left(\frac{1}{2} \sum_{k=1}^N (y_k - t_k)^2 \right) \\ &= \frac{\partial}{\partial \omega_{ik}} \left(\frac{1}{2} \sum_{k=1}^N [g(\sum_{i=0}^L \omega_{ik} x_i) - t_k]^2 \right) \\ &= \frac{1}{2} \sum_{k=1}^N 2(g(\sum_{i=0}^L \omega_{ik} x_i) - t_k) \frac{\partial}{\partial \omega_{ik}} \left(g(\sum_{i=0}^L \omega_{ik} x_i) - t_k \right) \\ &= \sum_{k=1}^N (y_k - t_k) \frac{\partial g}{\partial \omega_{ik}} x_i = \quad \text{Error} \times \text{slope} \times \text{input}\end{aligned}$$

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)

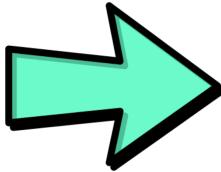
Cases 1:



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

Look at errors closer
(assume inputs are positive)

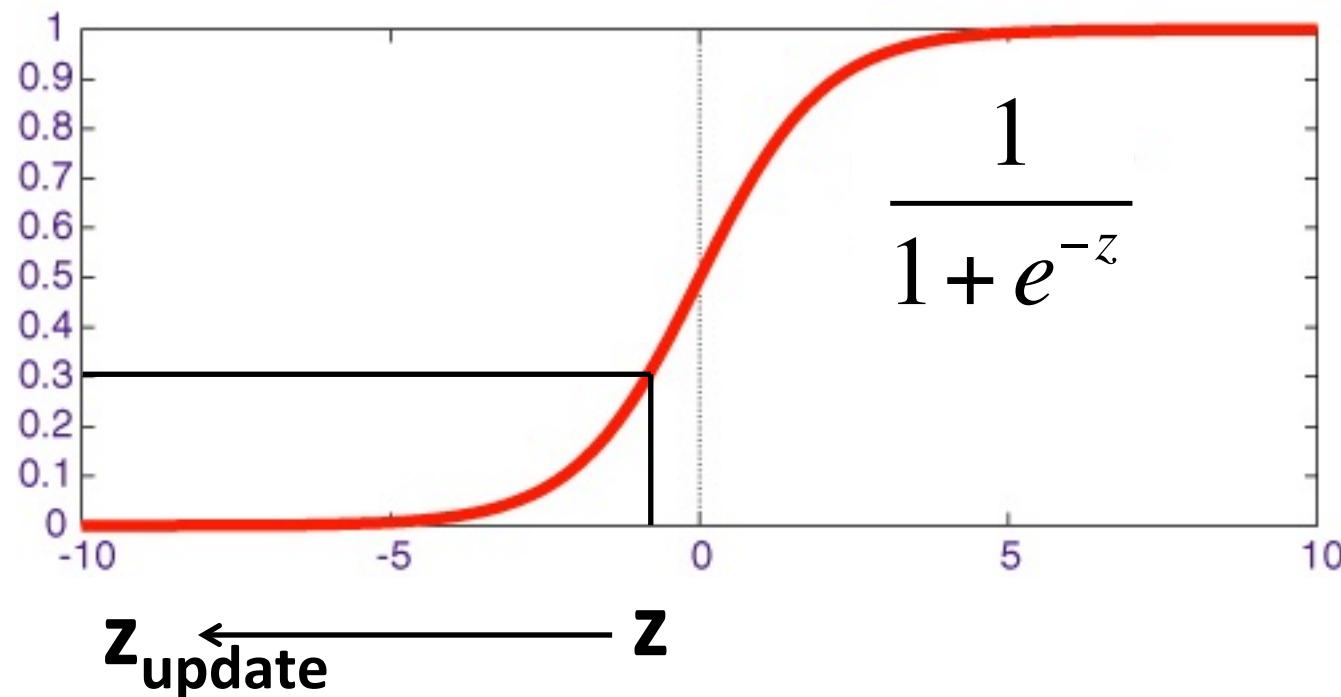
Cases 1:

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

Look at errors closer
(assume inputs are positive)

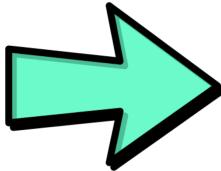
Cases 1:

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



Look at errors closer
(assume inputs are positive)

Cases 1:

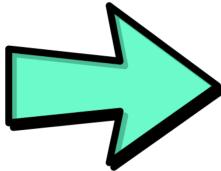
- Target is 0
 - Estimation is 0.3
- 
- Error = $0 - 0.3 = -0.3$

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

We need **reduce weights!**

Look at errors closer
(assume inputs are positive)

Cases 1:

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

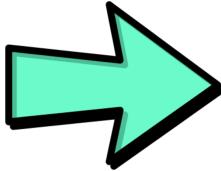
$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

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 = $0 - 0.3 = -0.3$

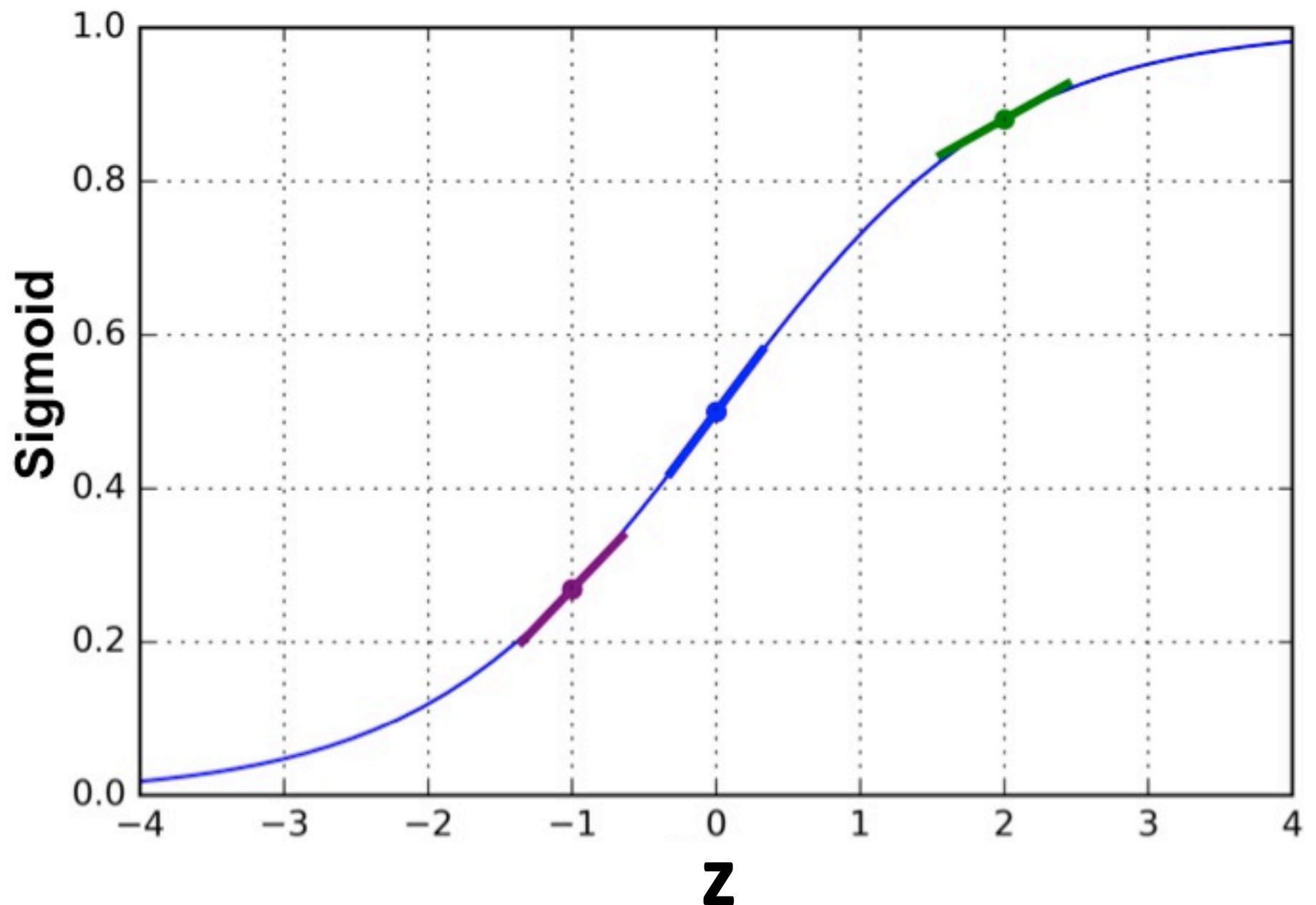
$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

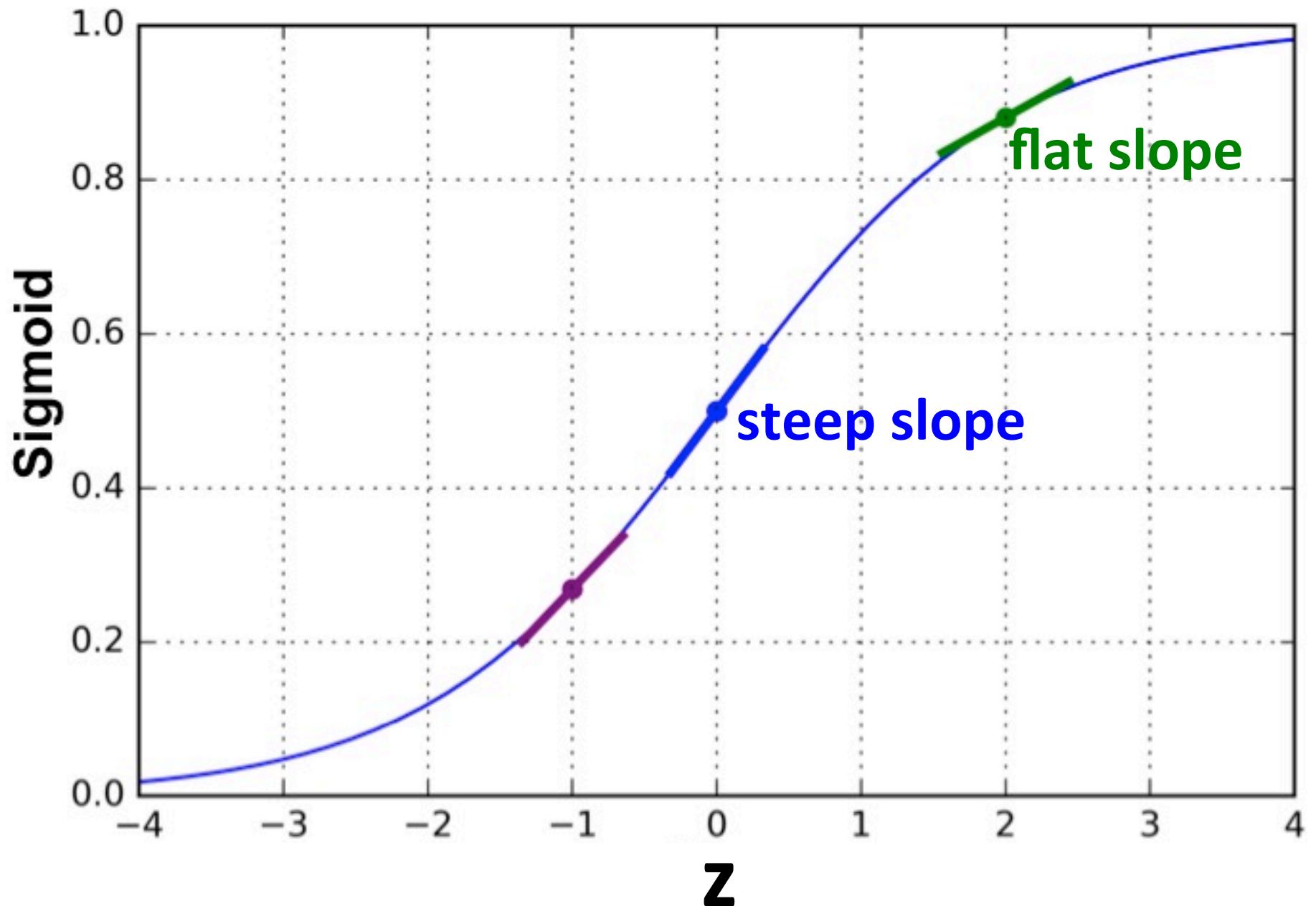
We need **reduce weights!**

But what if the inputs are negative

Weights update rules

Weights Delta = Error × input



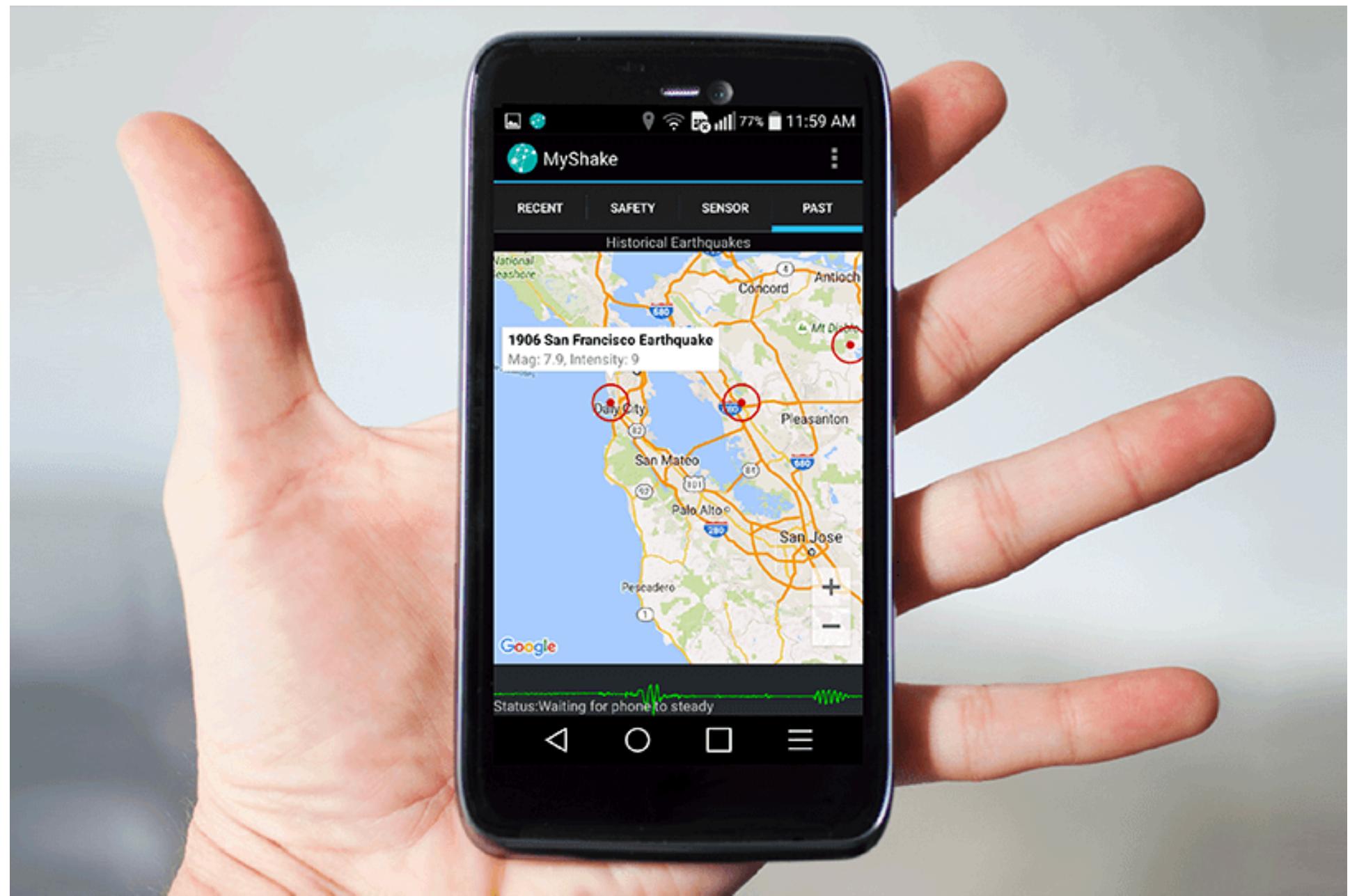


Weights update rules

Weights Delta = Error × slope × input

Learn from example



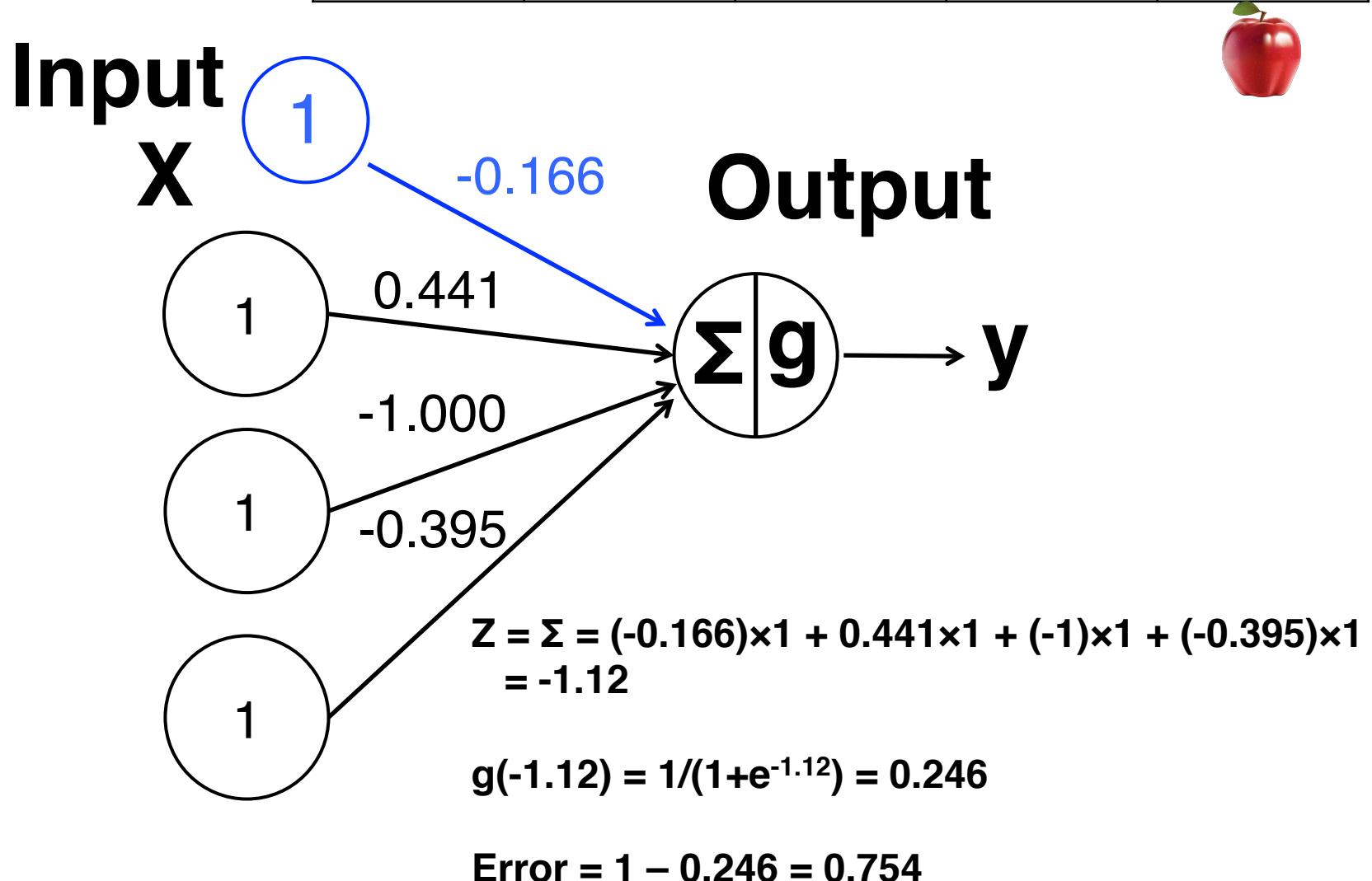


Learn from Example

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	0	1	0	1 

How to deal with errors

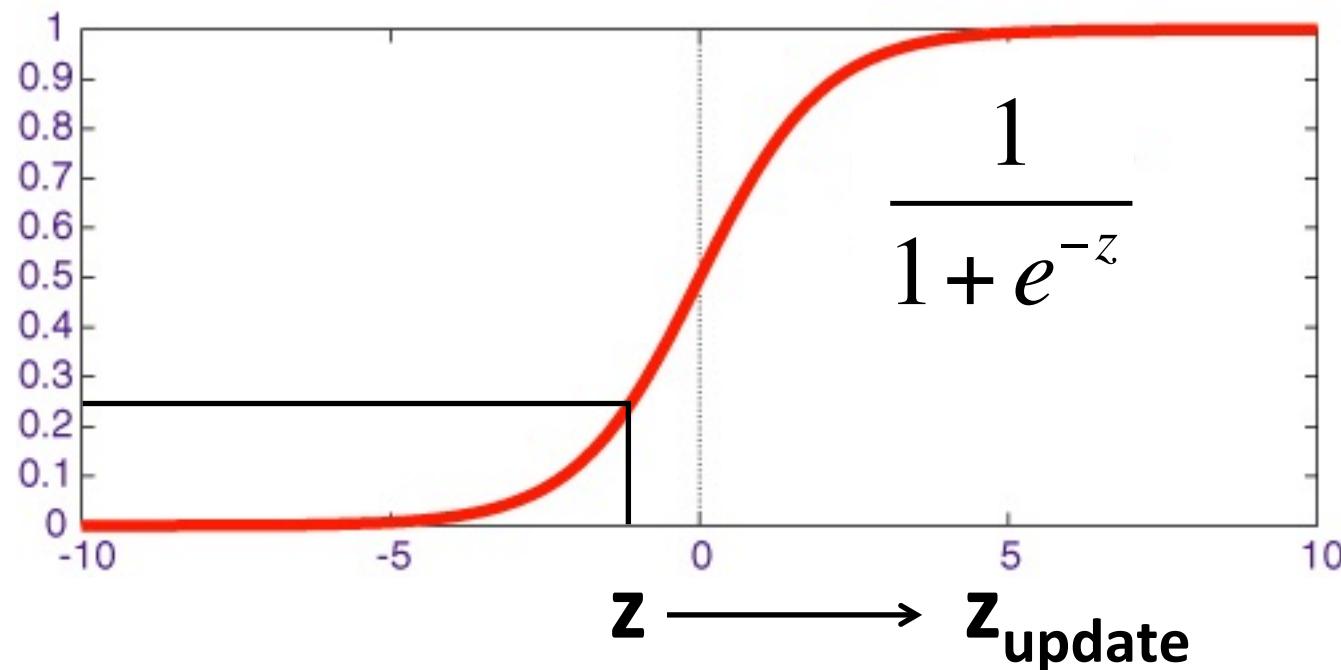
Sample	Feature 1	Feature 2	Feature 3	Target
Sample 1	1	1	1	1



- Target: 1
- Estimation: 0.246



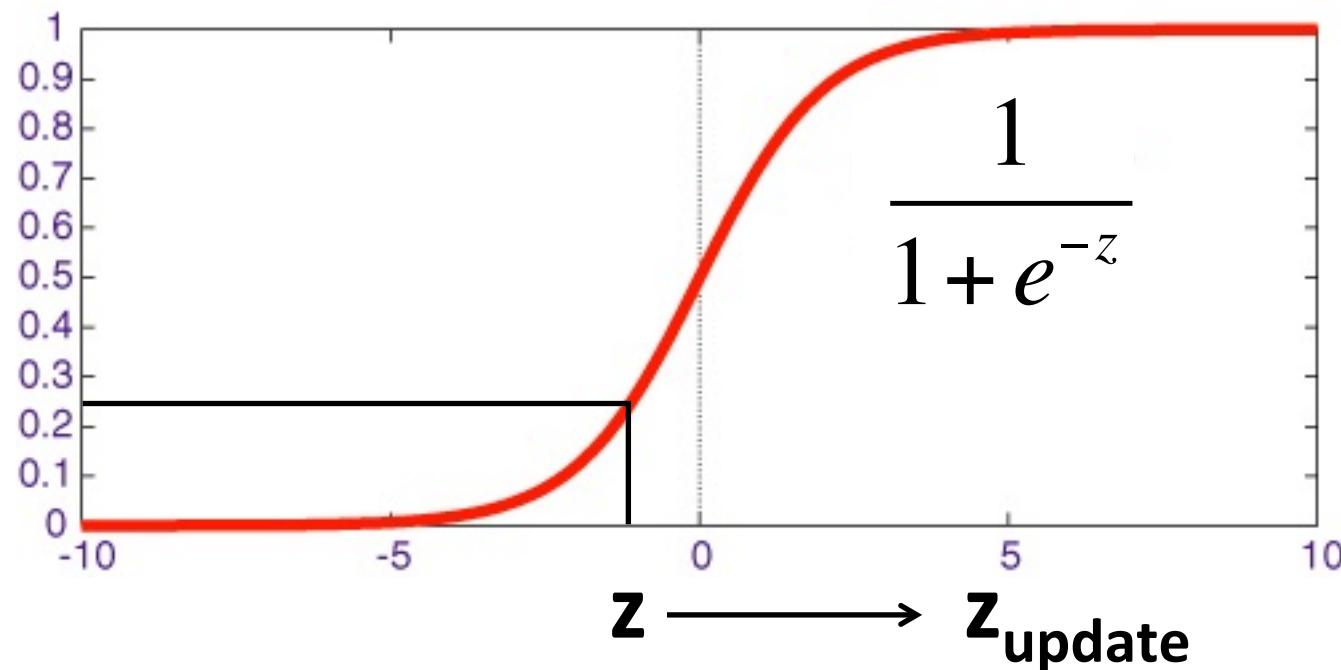
Error 0.754



- Target: 1
- Estimation: 0.246

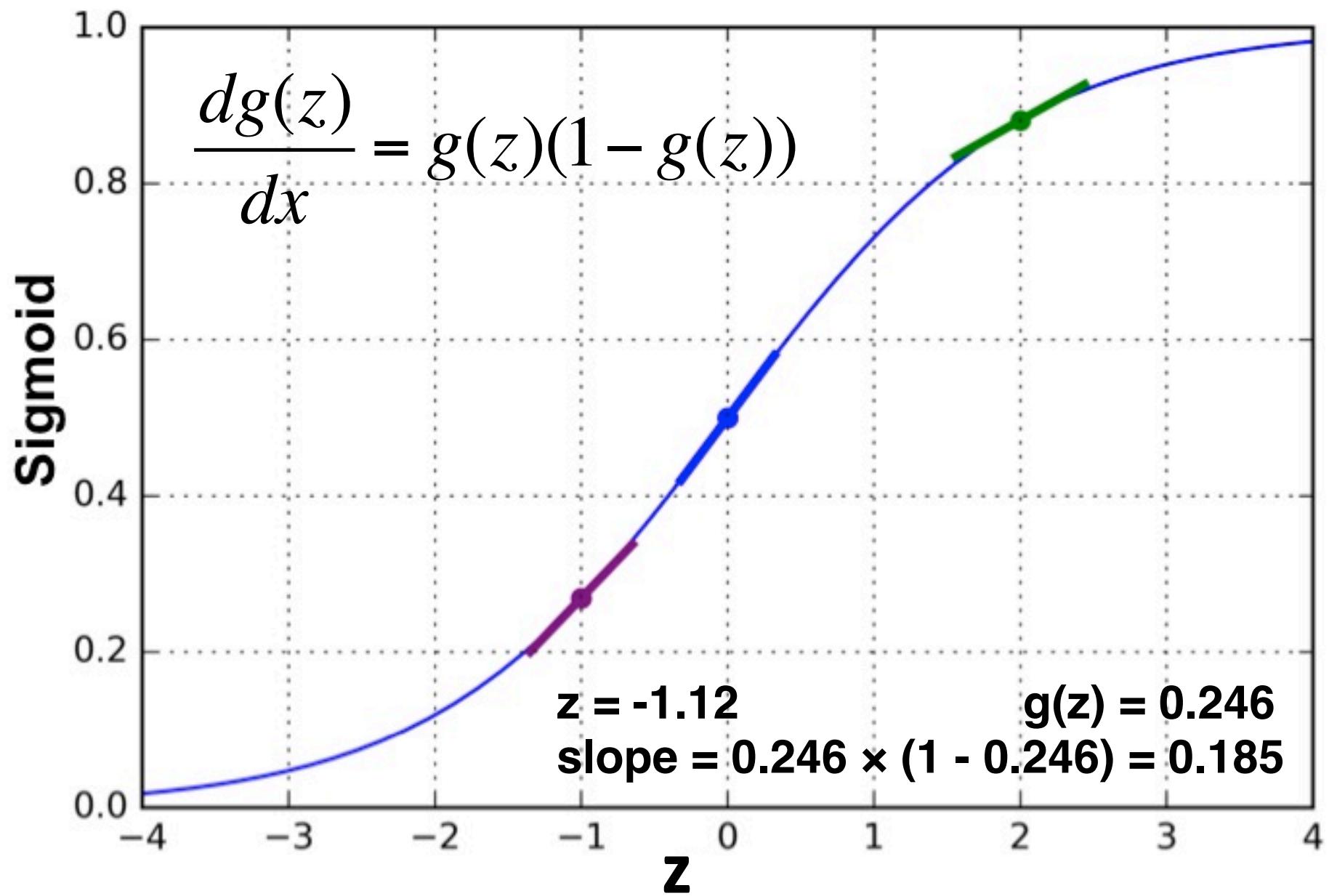


Error 0.754



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

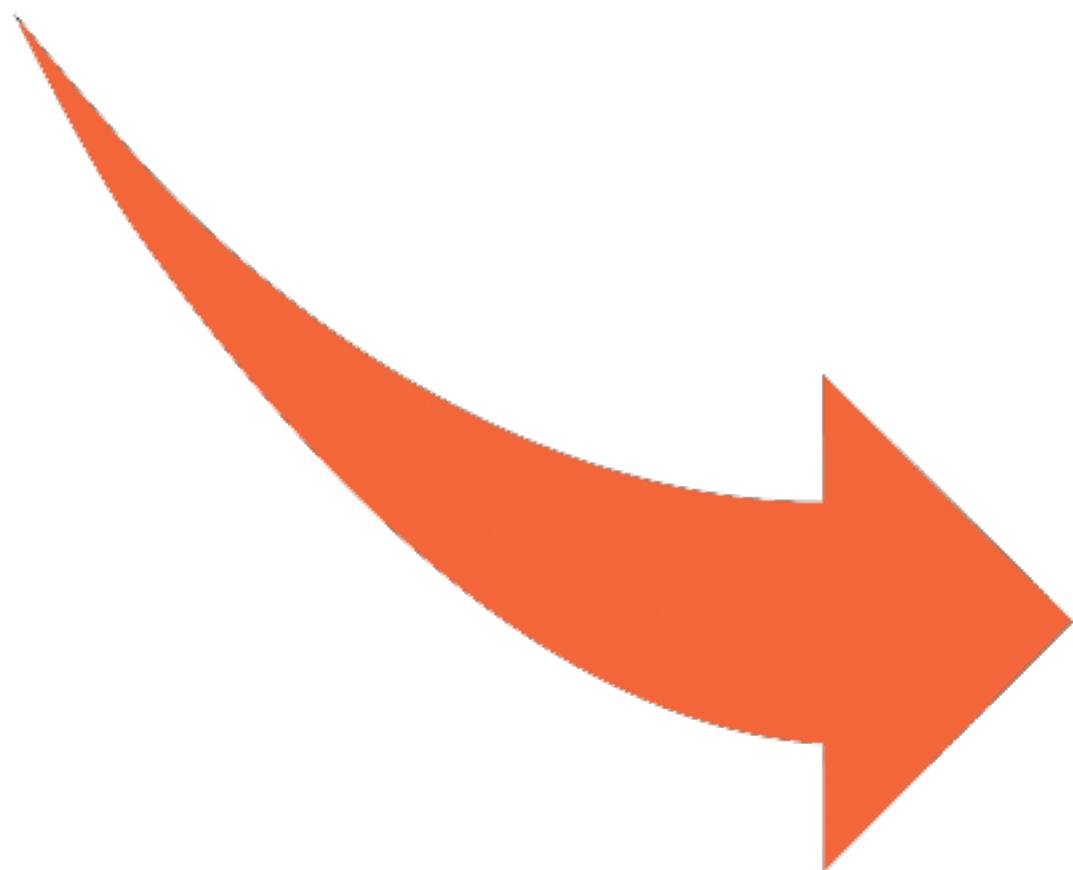
$$z = (-0.027) \times 1 + 0.580 \times 1 + (-0.861) \times 1 + (-0.256) \times 1 = -0.564$$

$$g(z) = 0.637$$

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

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.



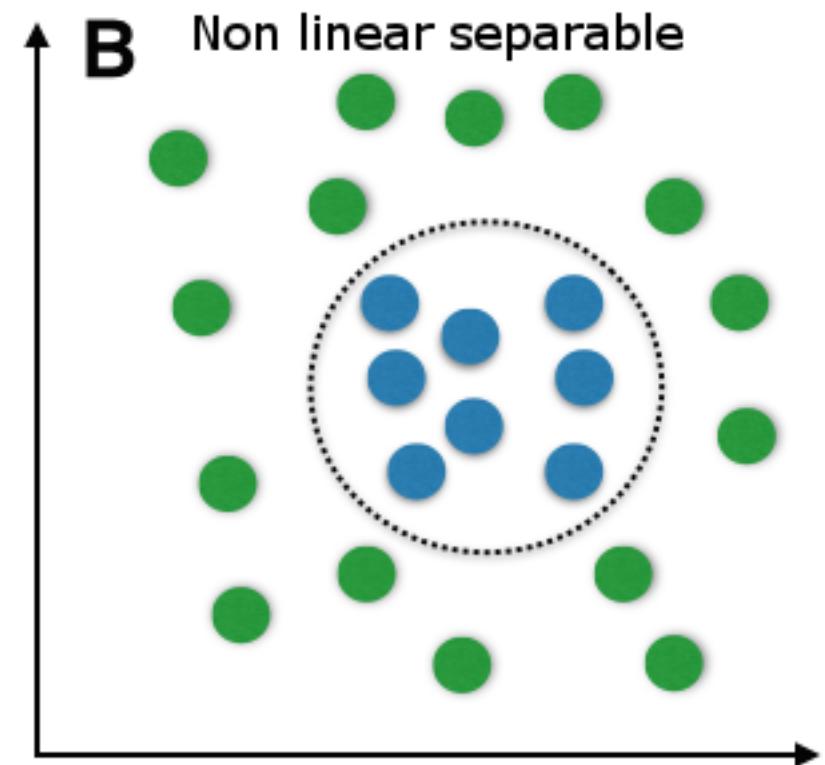
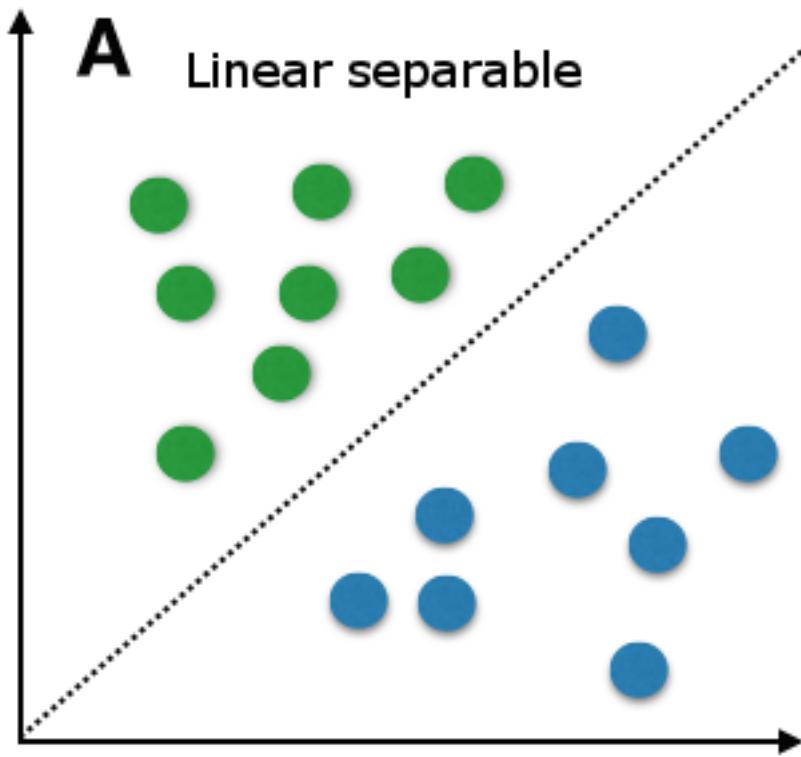
80



152

...

Perceptron limitations

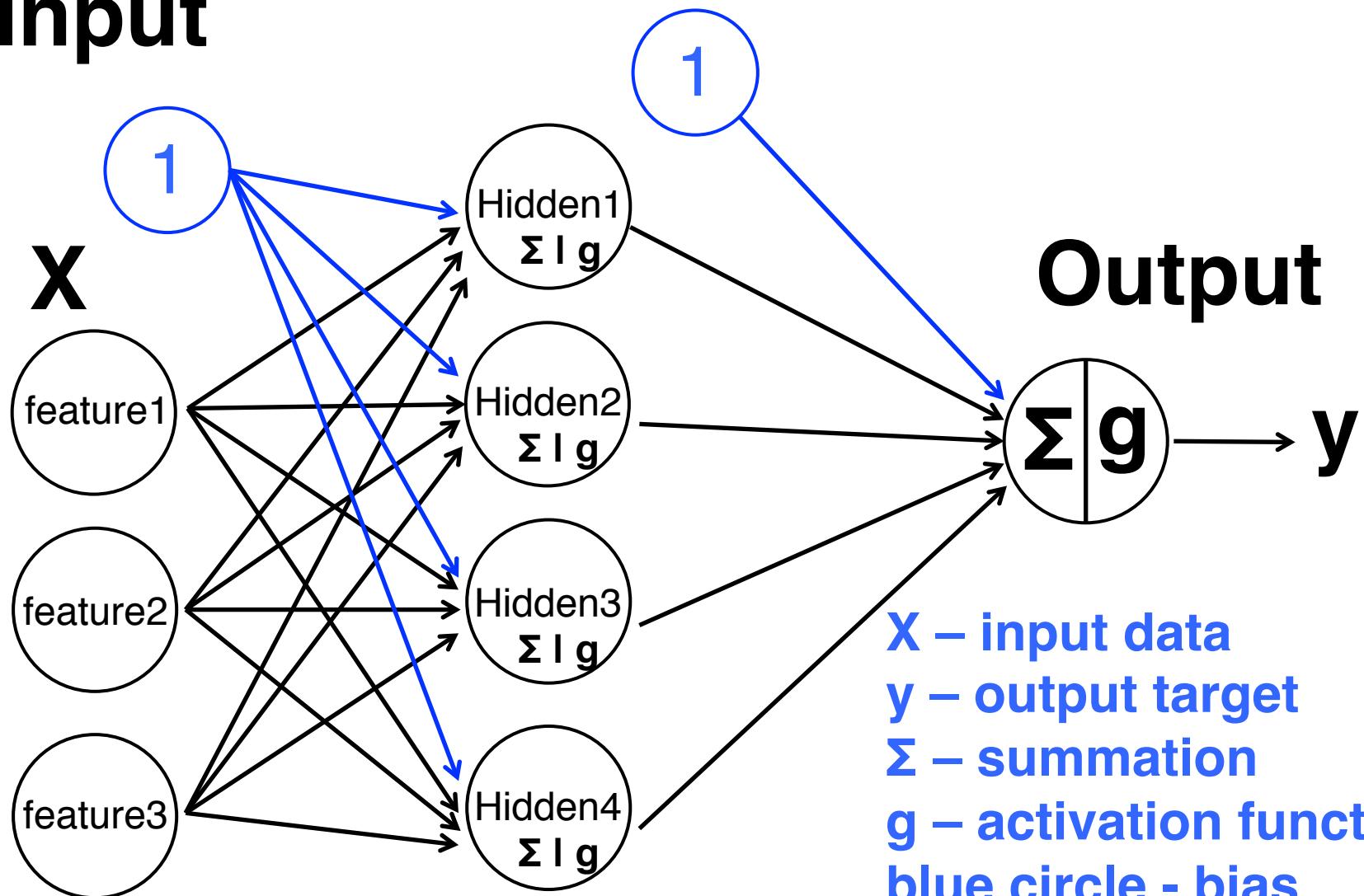


Winter of ANN



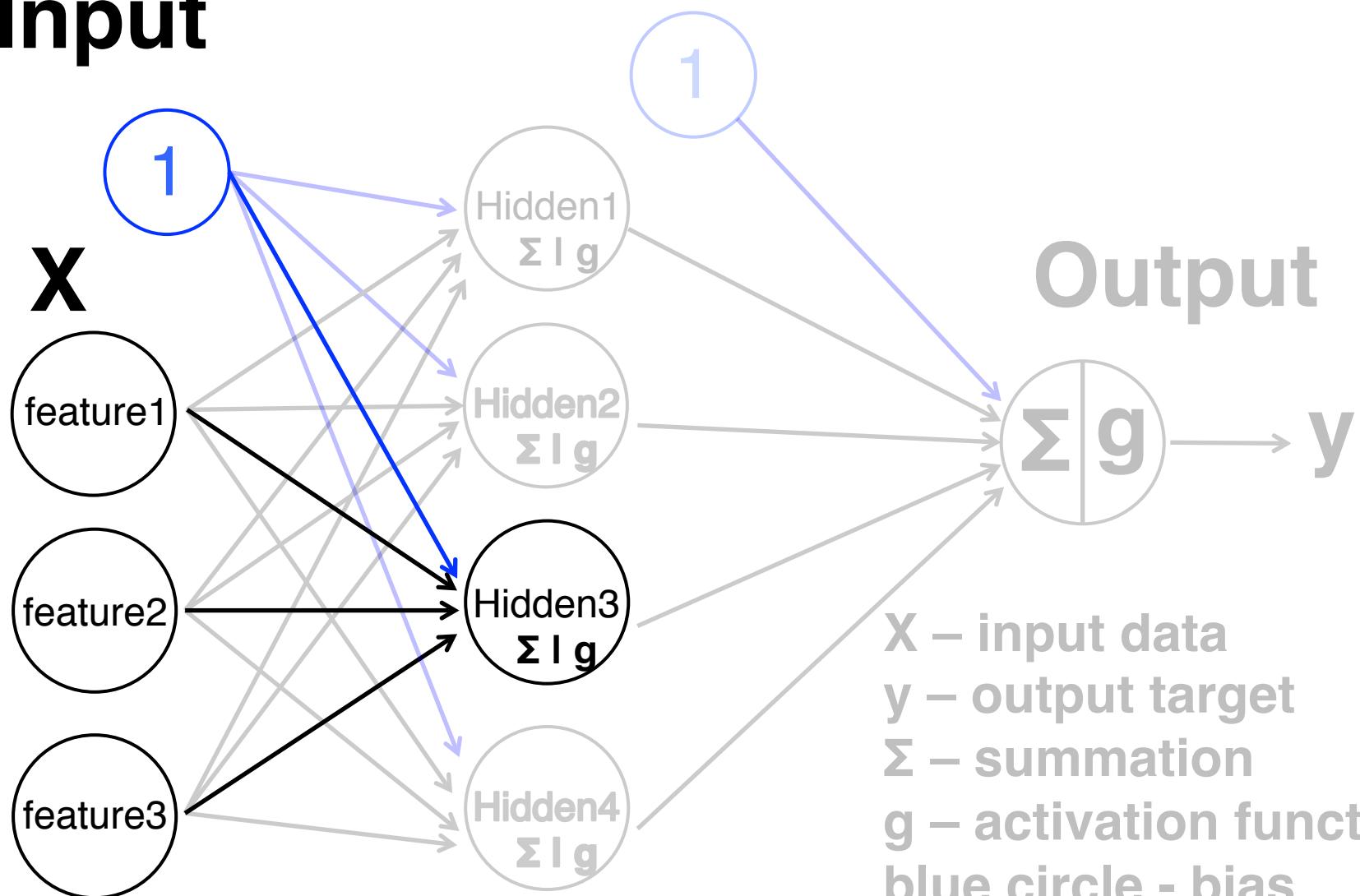
Multi-Layer Perceptron

Input



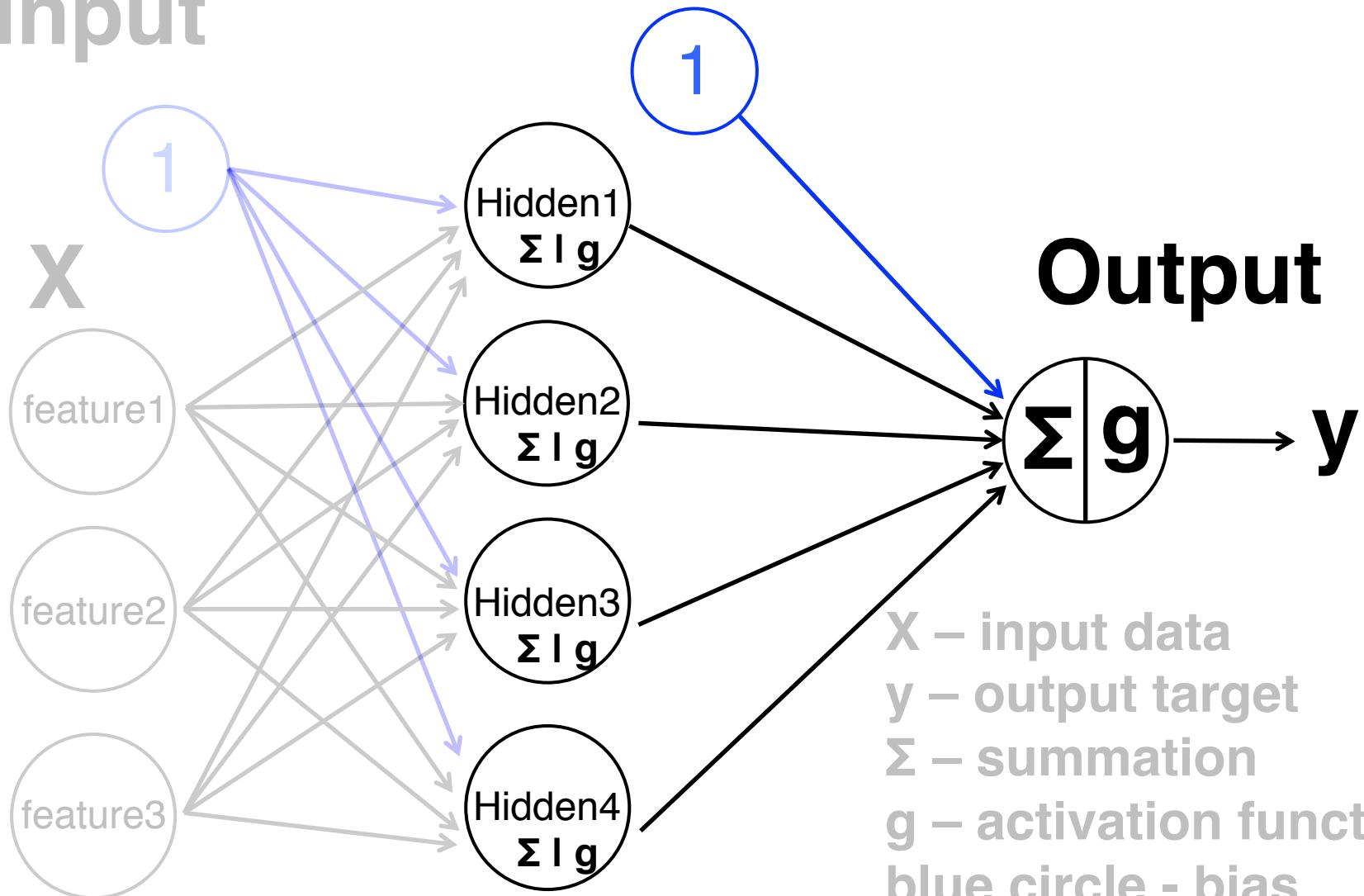
X – input data
y – output target
 Σ – summation
g – activation function
blue circle - bias

Input



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

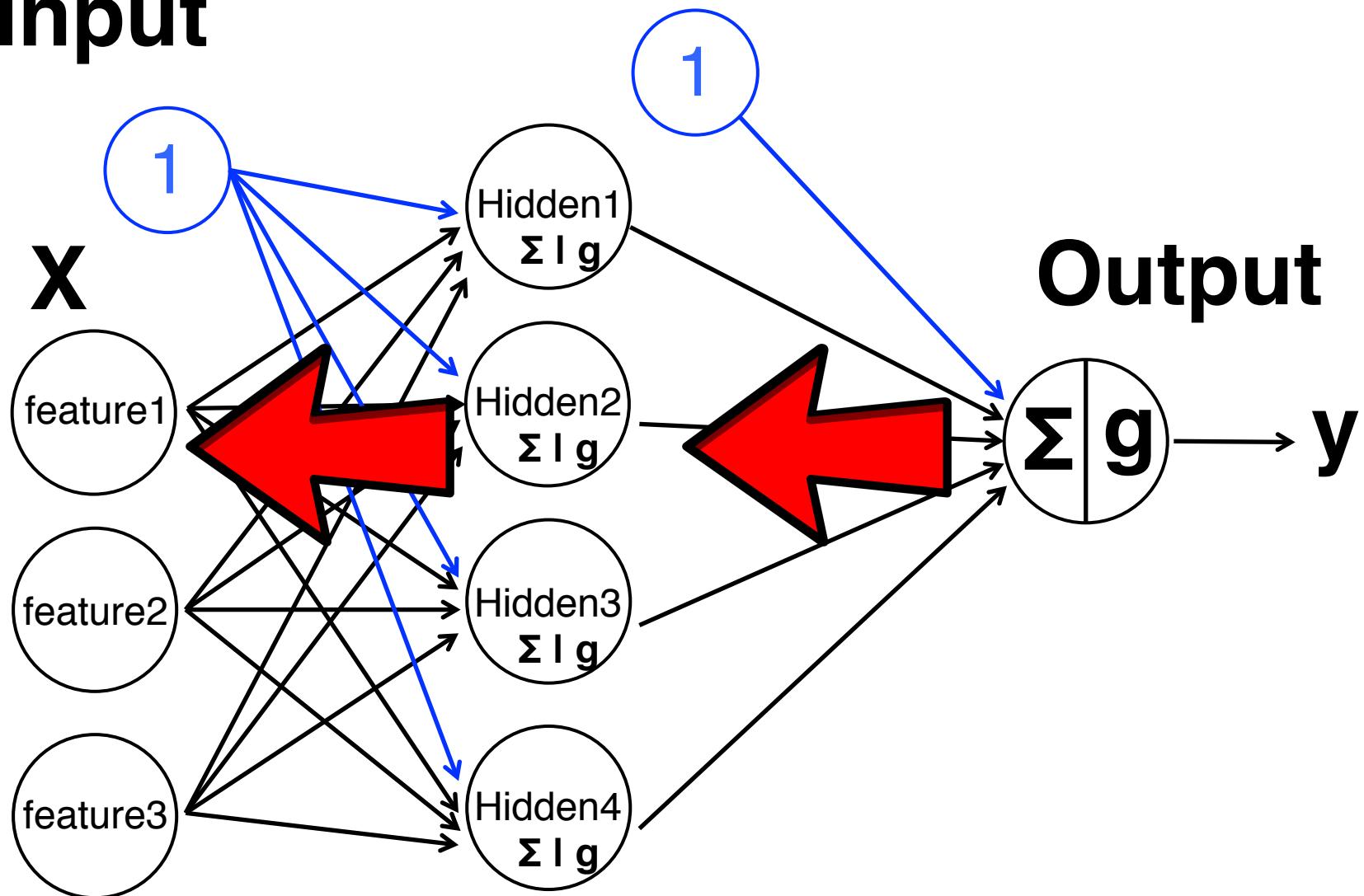
Input



Output

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

Input



Output

y

Go to notebook 02

Learning curve

Workshop time

Gentle introduction

- What's ML
- ANN history
- ANN overview

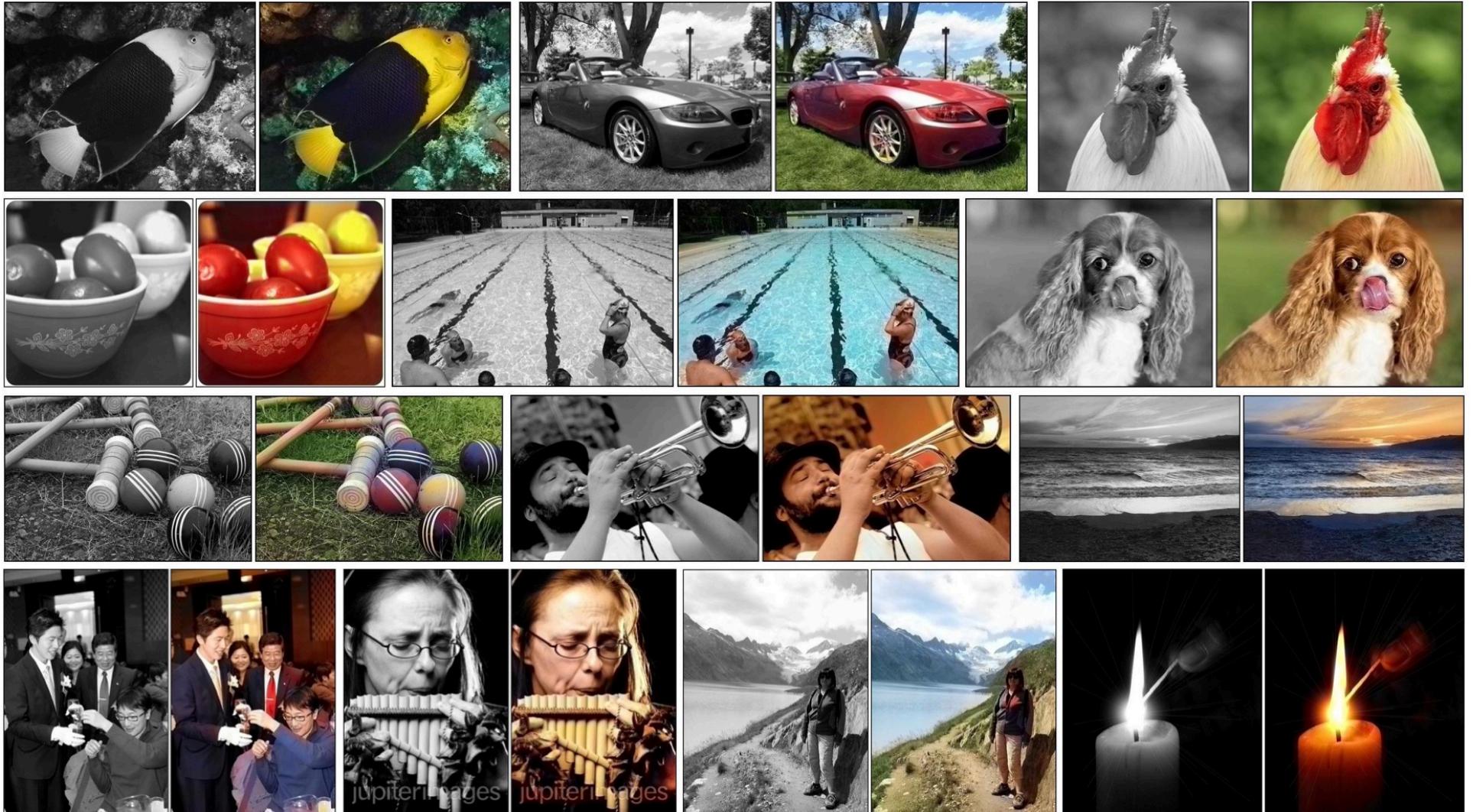
Step by step ANN

- 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

Fifty One &/100

DOLLARS



THE FIFTH THIRD BANK

CINCINNATI, OHIO 45201

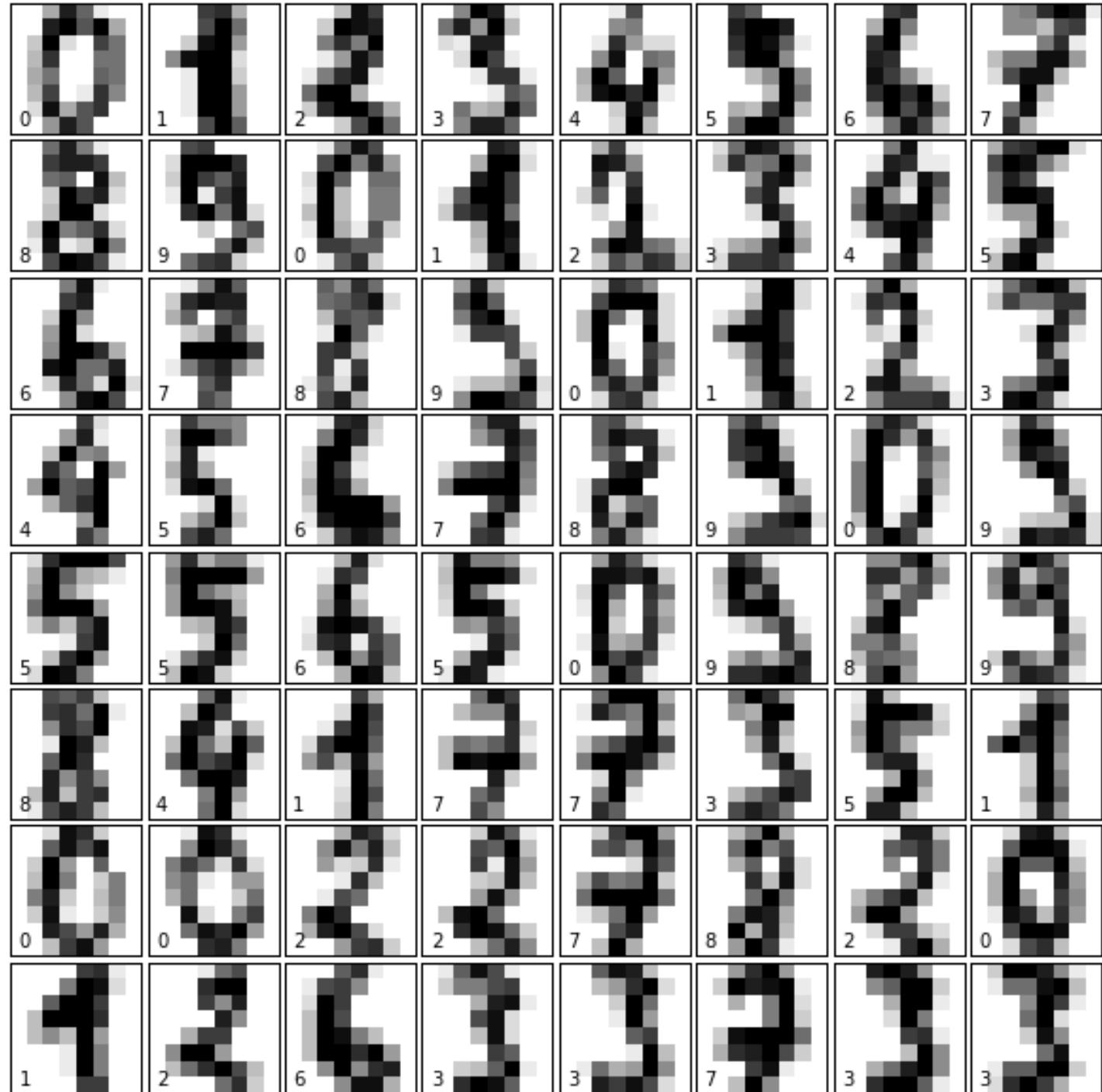
FIFTH THIRD CENTER
38 FOUNTAIN SQUARE PLAZA, CINCINNATI, OH 45263

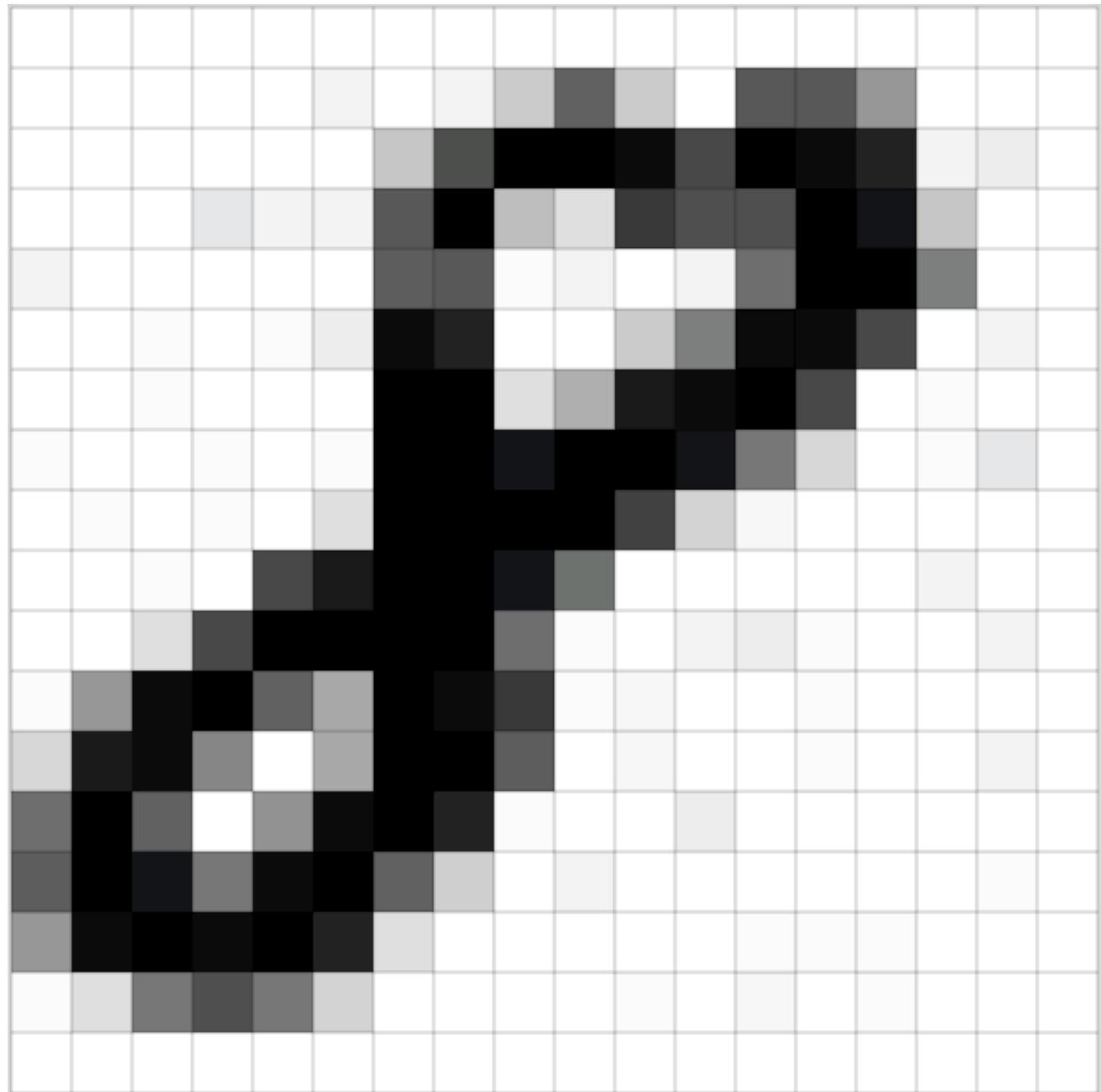
FOR RMB

1042000341 2214 590 53609

00000005000.00

N. A. Armstrong





Go to notebook 03

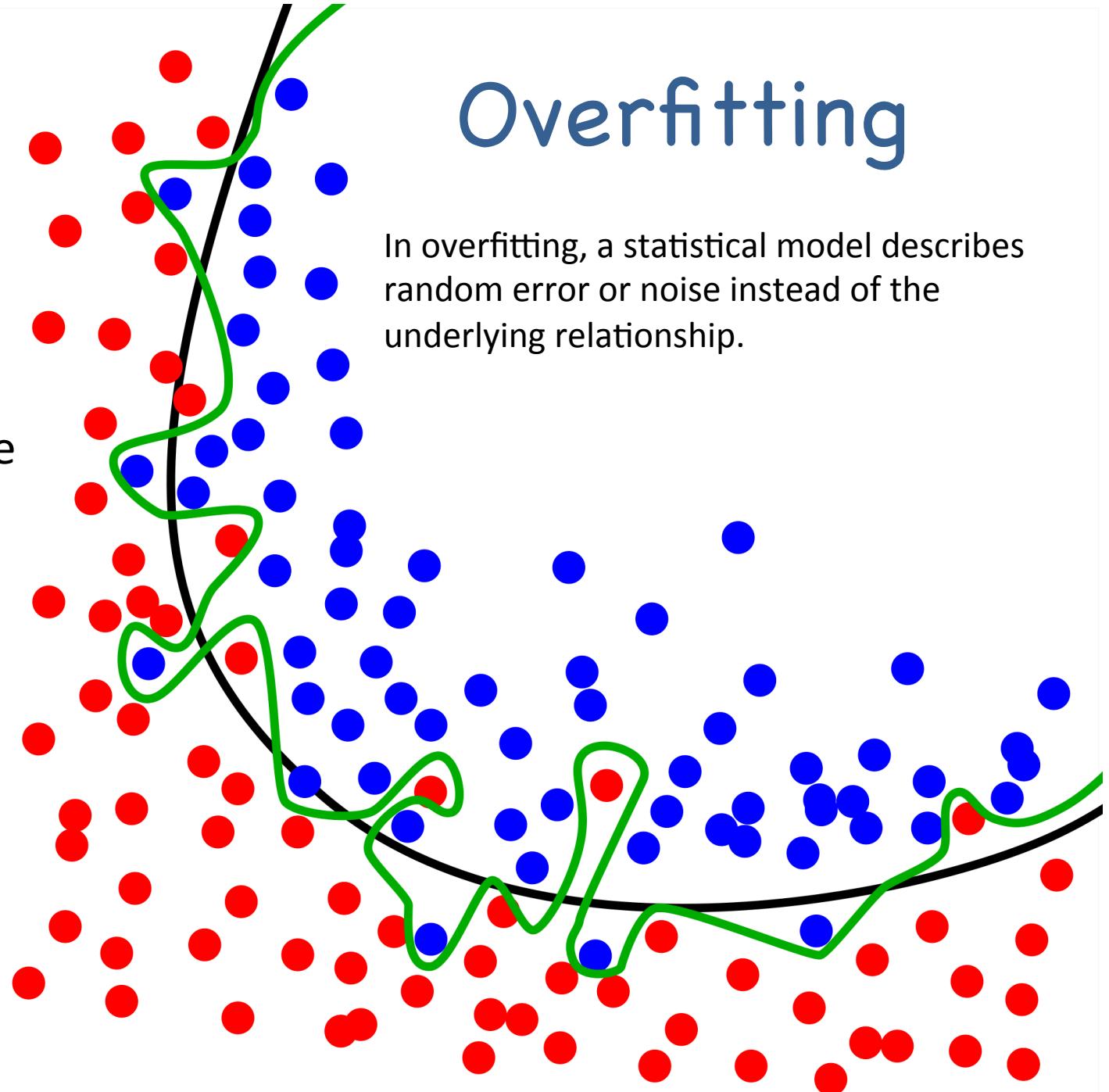
More on ANN



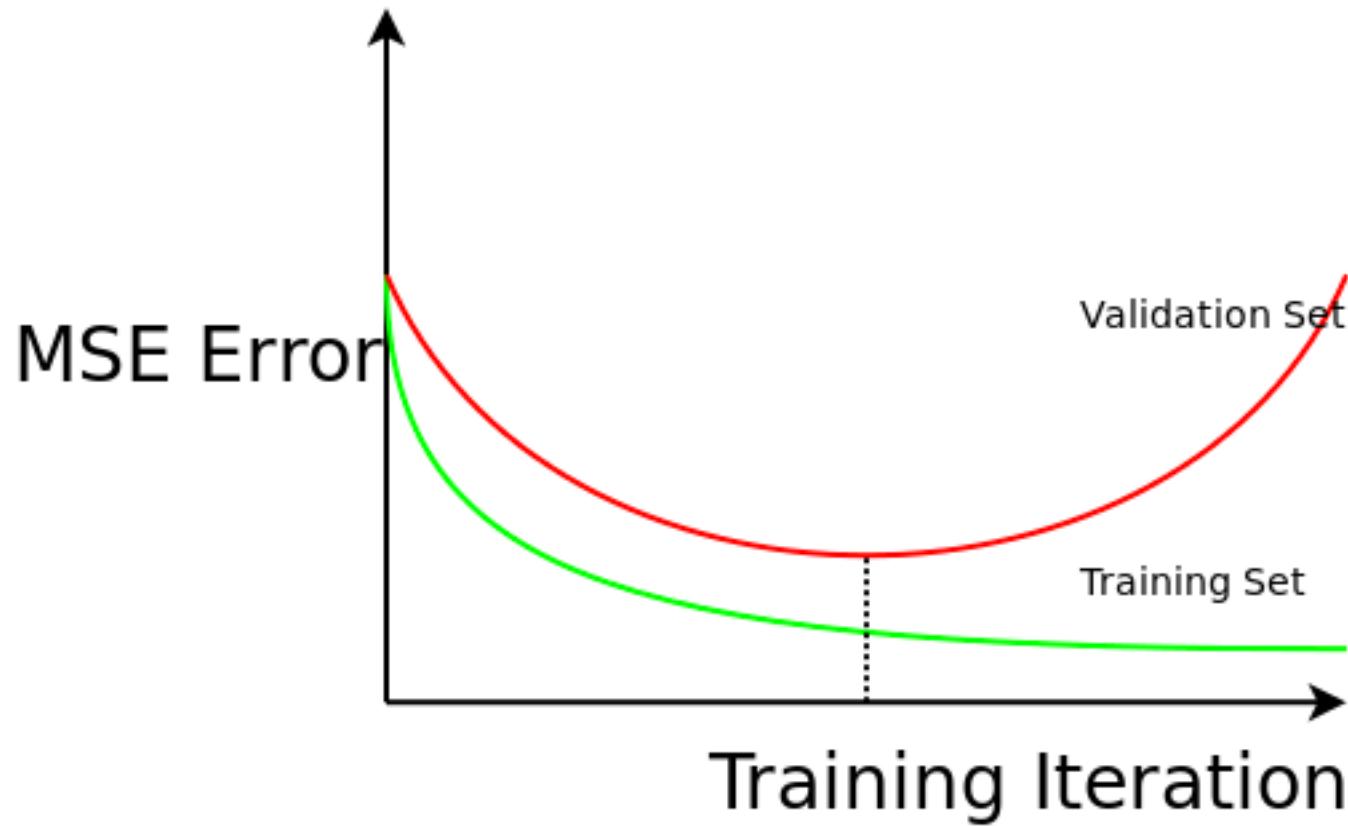
Overfitting

Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

In overfitting, a statistical model describes random error or noise instead of the underlying relationship.



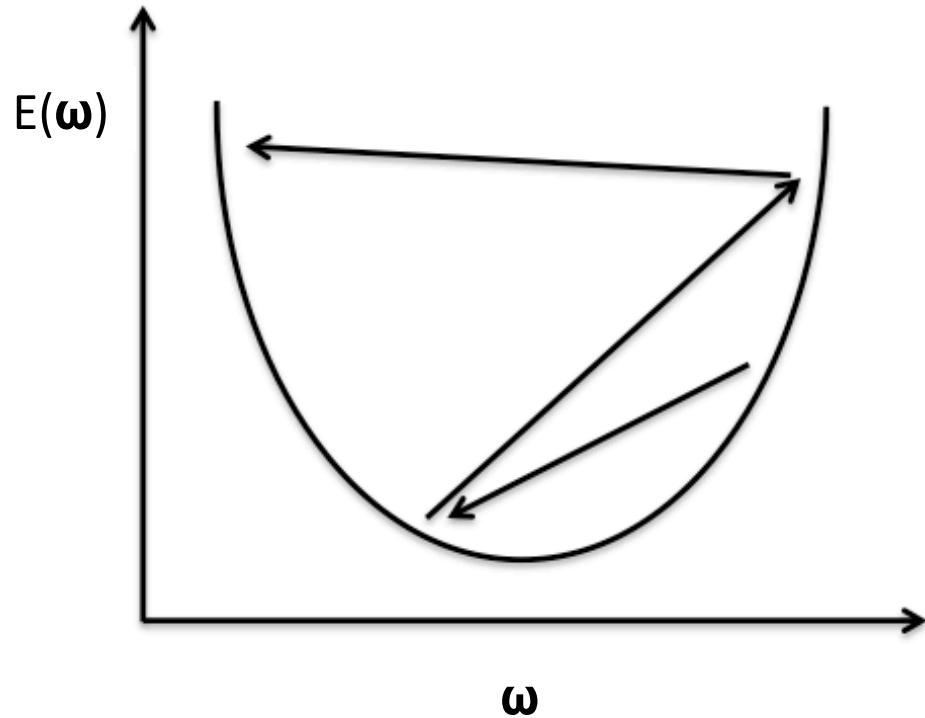
When to stop



Training error is the error that you get when you run the trained model back on the training data.

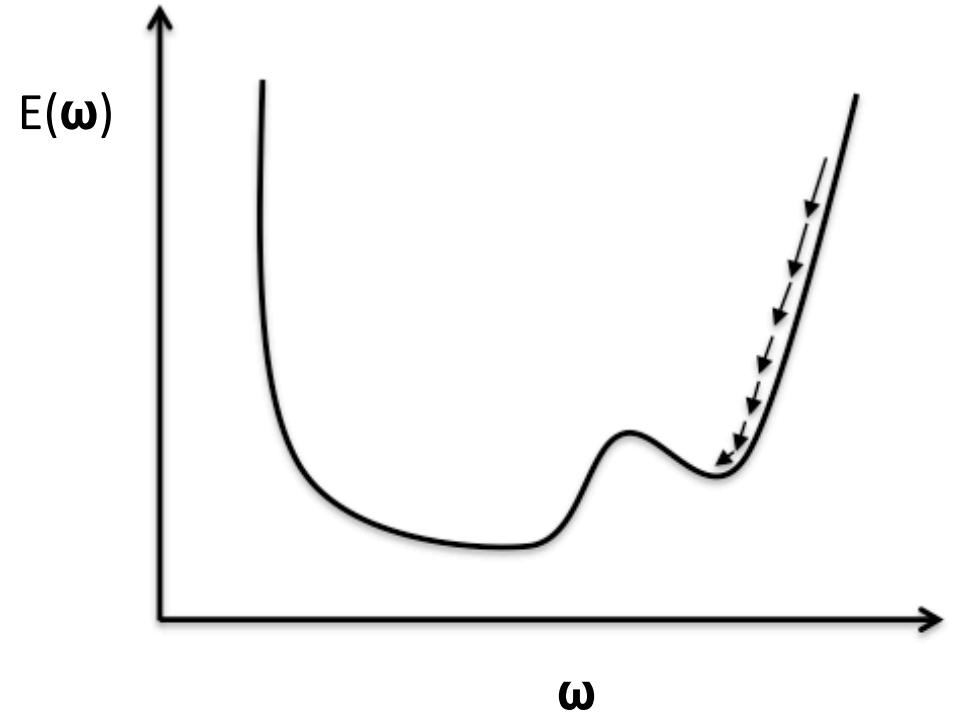
Validation error is the error when you get when you run the trained model on a set of data that it has previously never been exposed to.

Learning rate



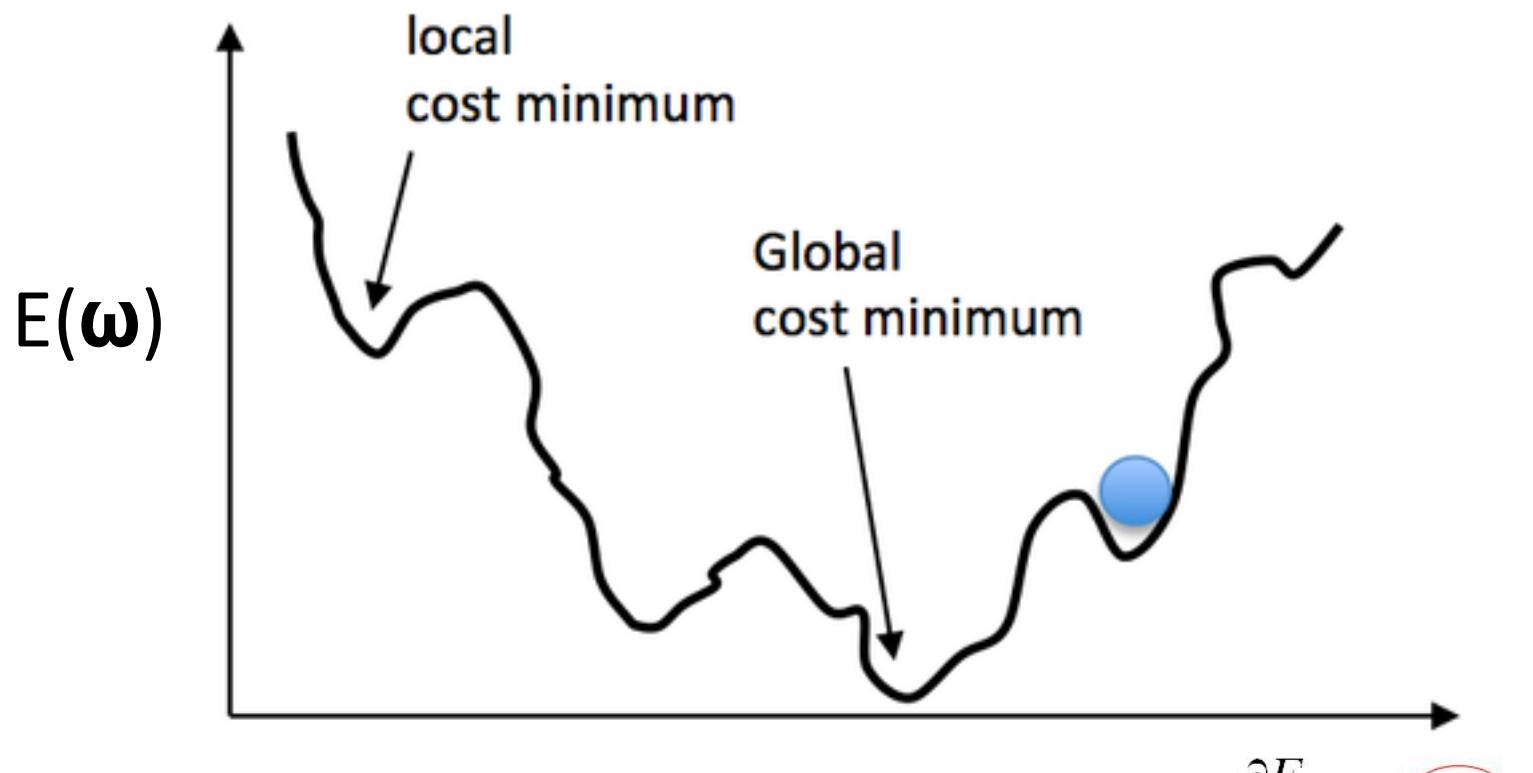
Large learning rate: Overshooting.

Learning rate controls step size of each iteration. Use an adaptive learning rate is better.



Small learning rate: Many iterations until convergence and trapping in local minima.

Momentum



A simple way to avoid trapping into a local minimum. (Not guarantee to find the global minimum though)

$$\Delta w^t = -\eta \frac{\partial E}{\partial w} + \alpha \Delta w^{t-1}$$

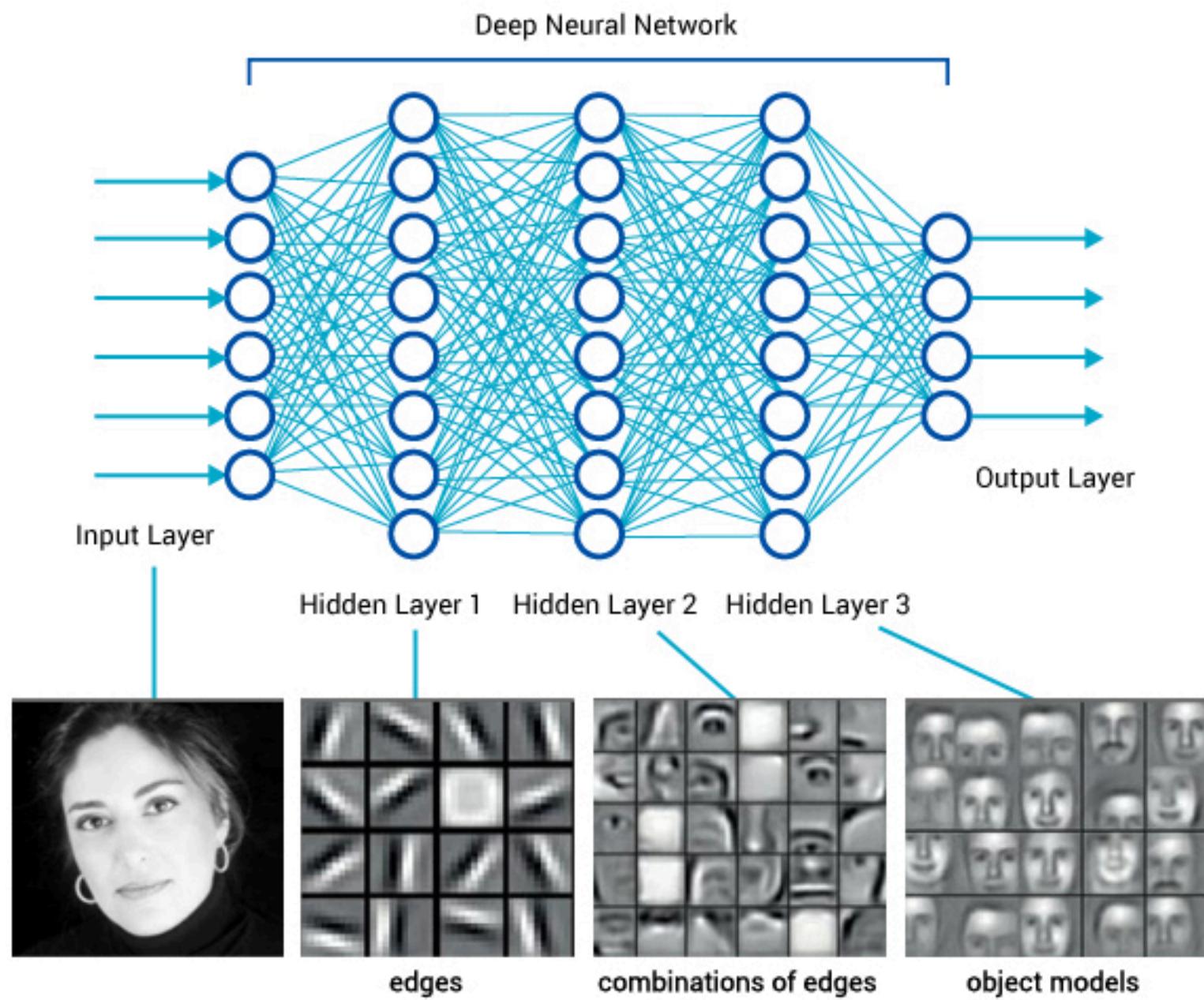
Diagram illustrating the momentum update rule:

- Momentum parameter 0.1-0.8*: Points to the coefficient α in the equation.
- Momentum term*: Points to the term $\alpha \Delta w^{t-1}$ in the equation.

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 ...



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>

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