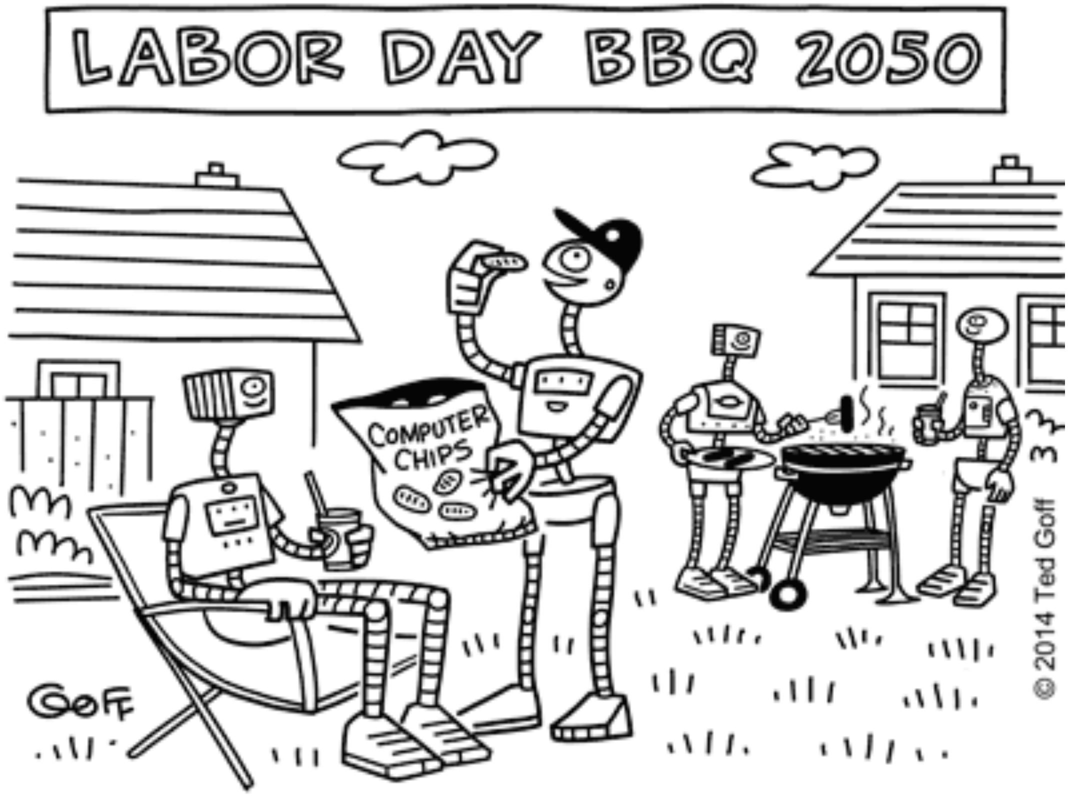


"Try one of these. They're salty, and they come with nine new human skills."

https://github.com/qingkaikong/20170306_ML_ANN_basics_DT

Deutsche Telekom, Inc.
Silicon Valley Innovation Center

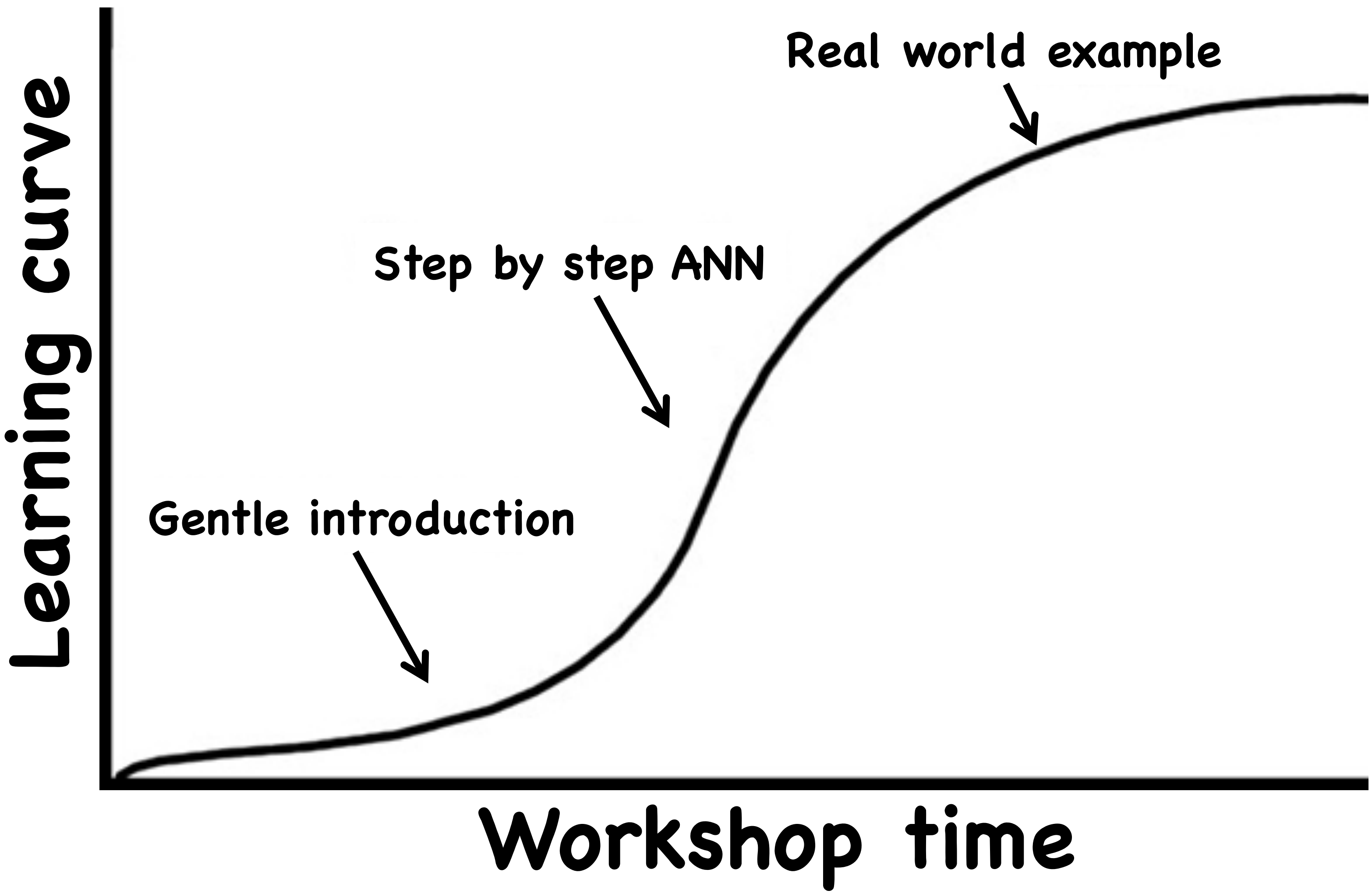
University of California Berkeley Seismological Laboratory

Machine learning – Artificial Neural Network basics

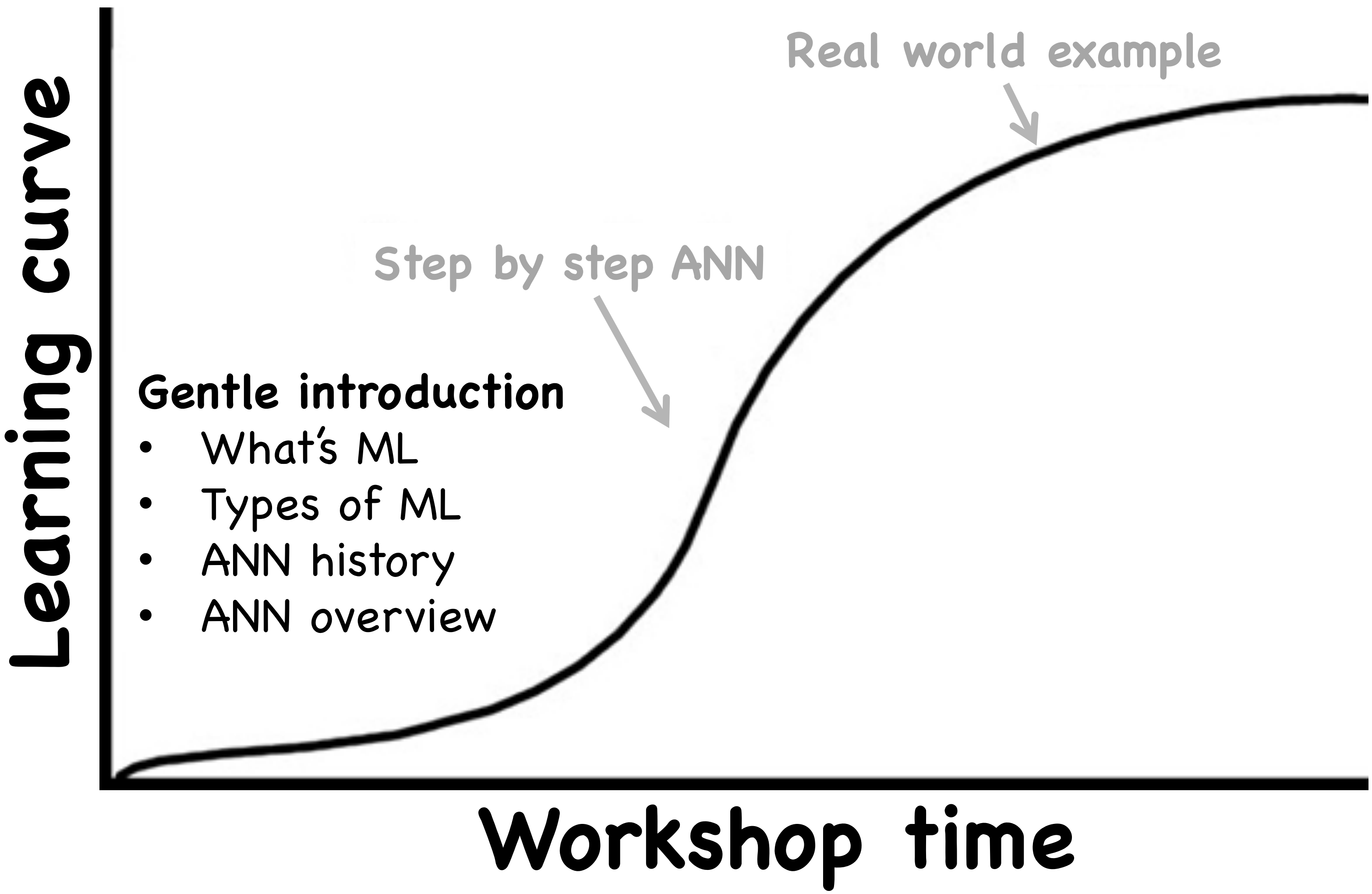
Qingkai Kong

2017-03-06

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

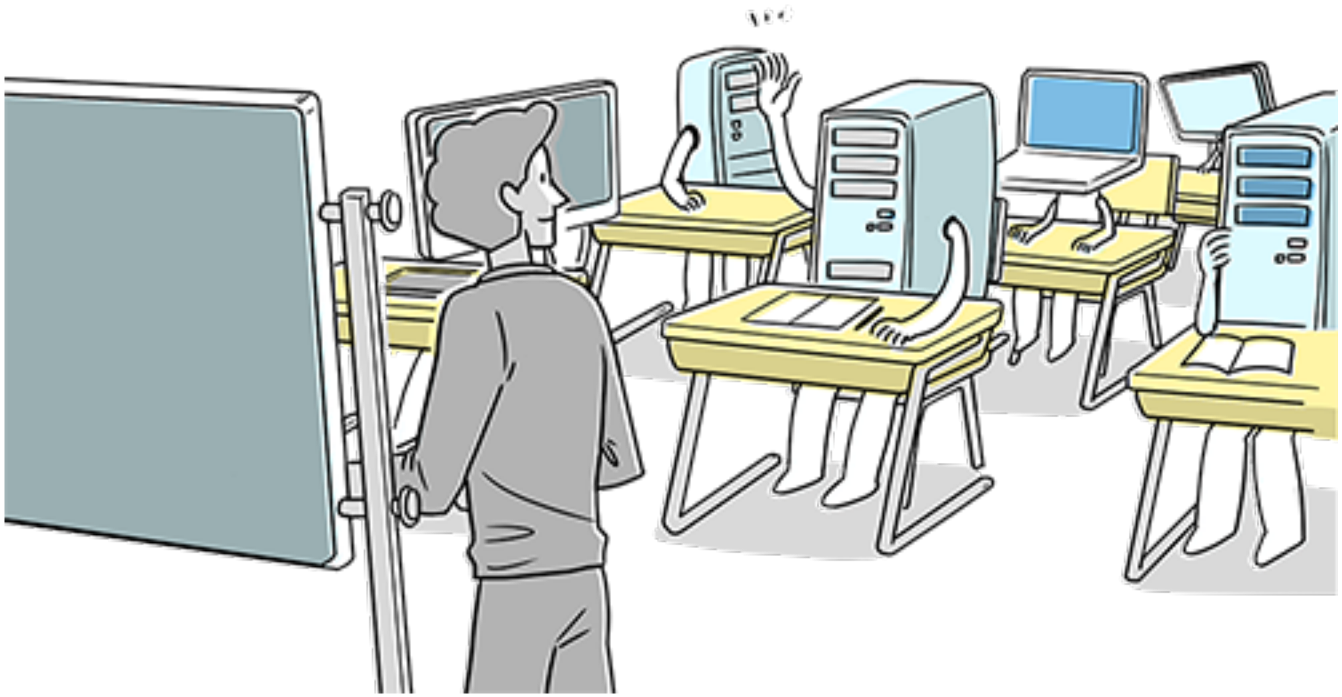


https://github.com/qingkaikong/20170306_ML_ANN_basics_DT

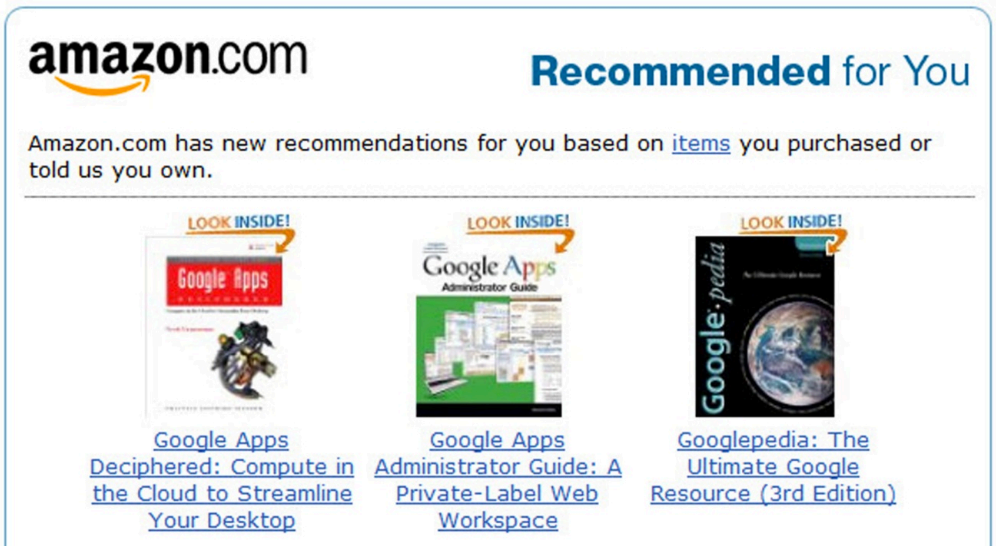


https://github.com/qingkaikong/20170306_ML_ANN_basics_DT

What is machine learning?



https://github.com/qingkaikong/20170306_ML_ANN_basics_DT



Self-driving car

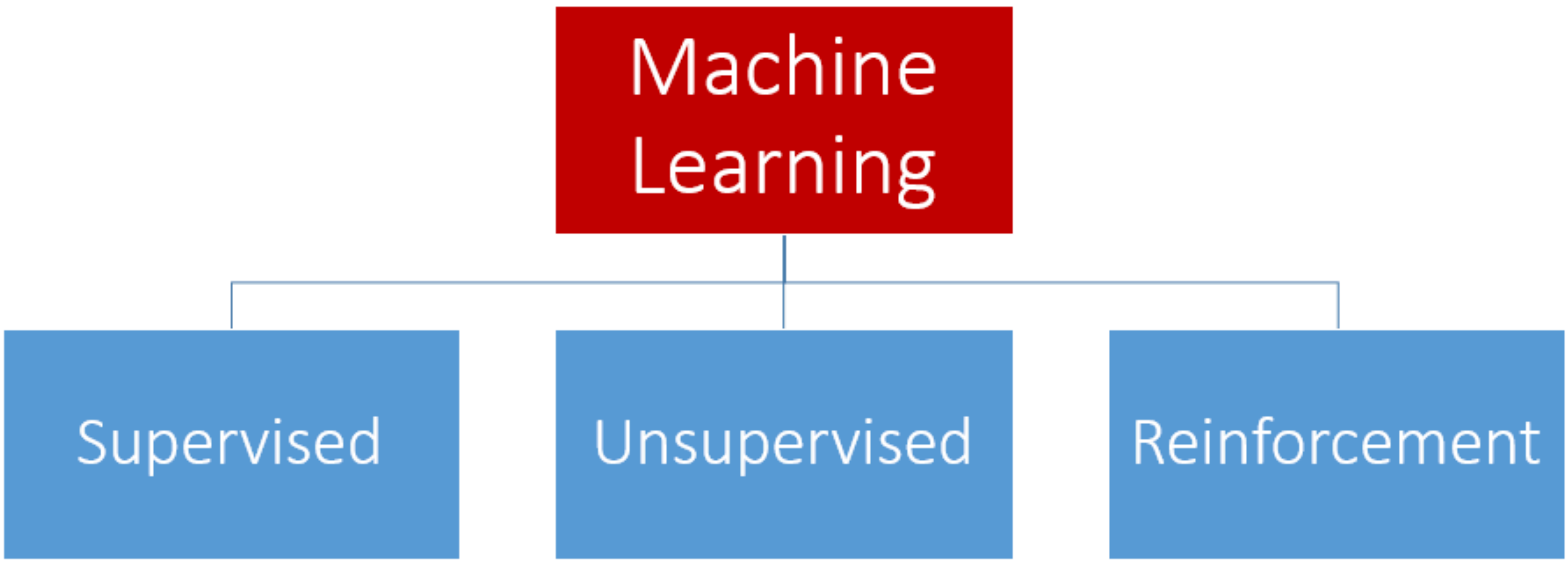
Voice recognition

...

https://github.com/qingkaikong/20170306_ML_ANN_basics_DT

Not always working

Types of Machine Learning



Task driven (Regression / Classification)

Data driven (Clustering)

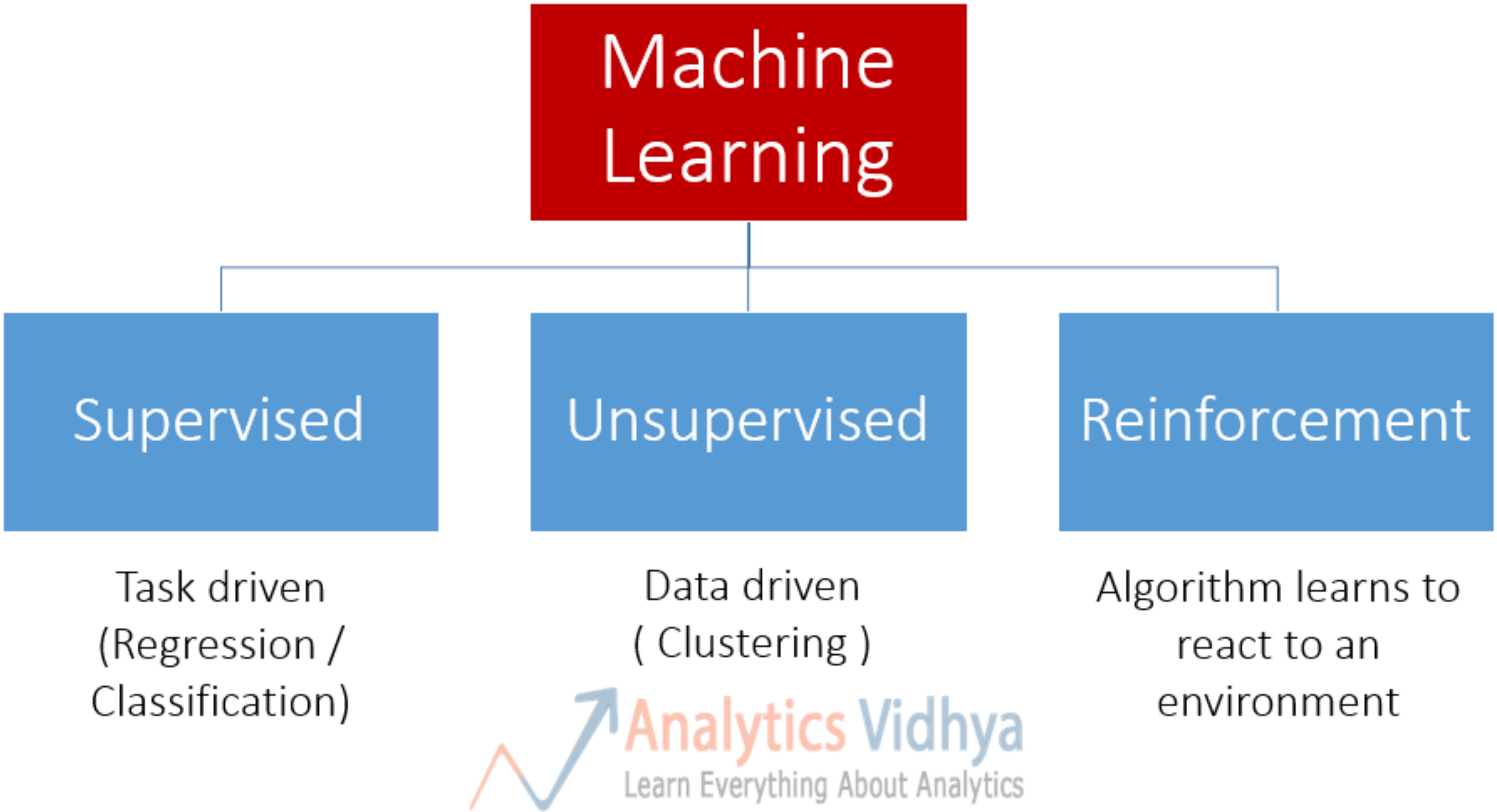
Algorithm learns to react to an environment

Supervised learning

Unsupervised learning

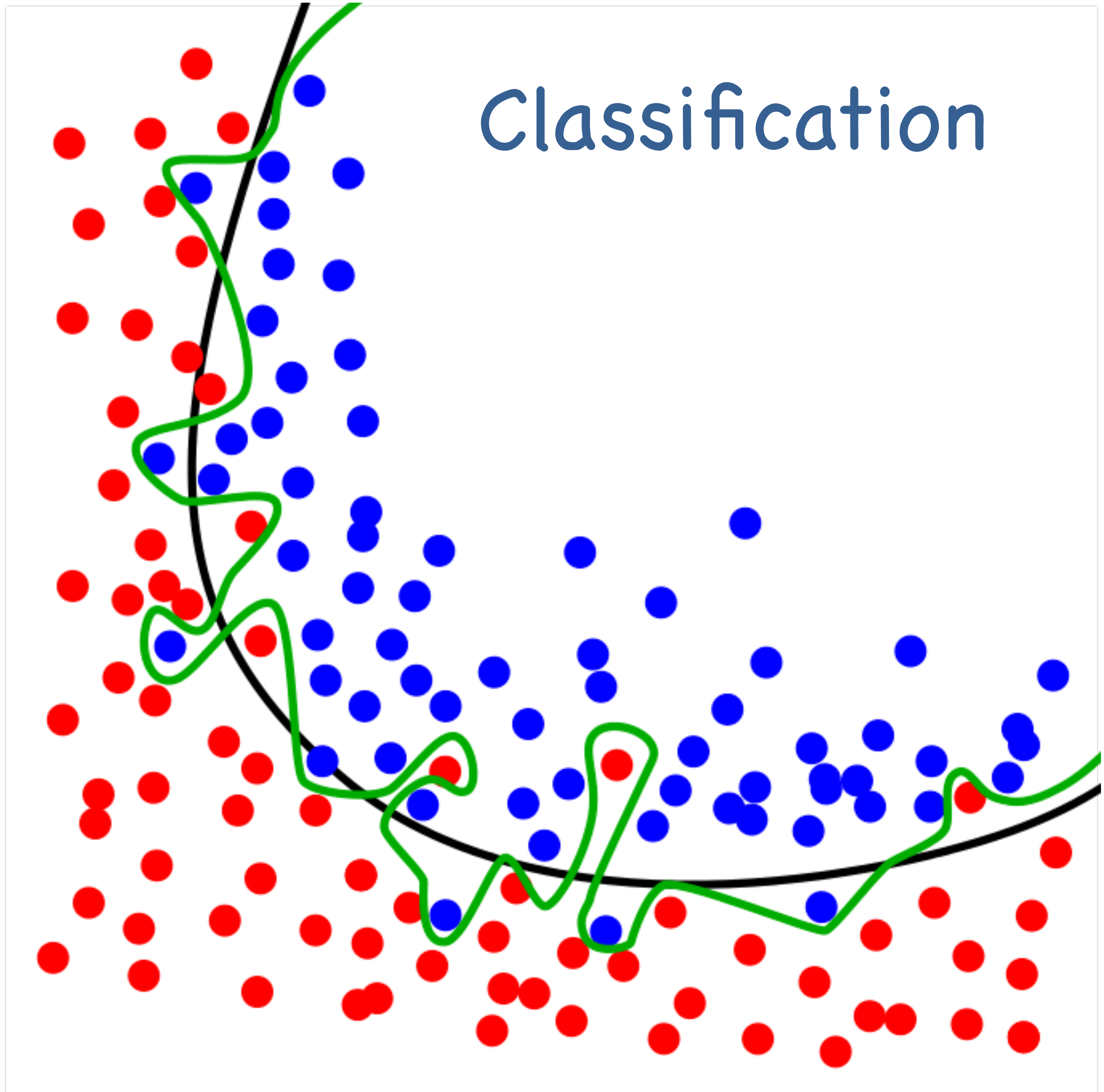
Reinforcement learning

Types of Machine Learning

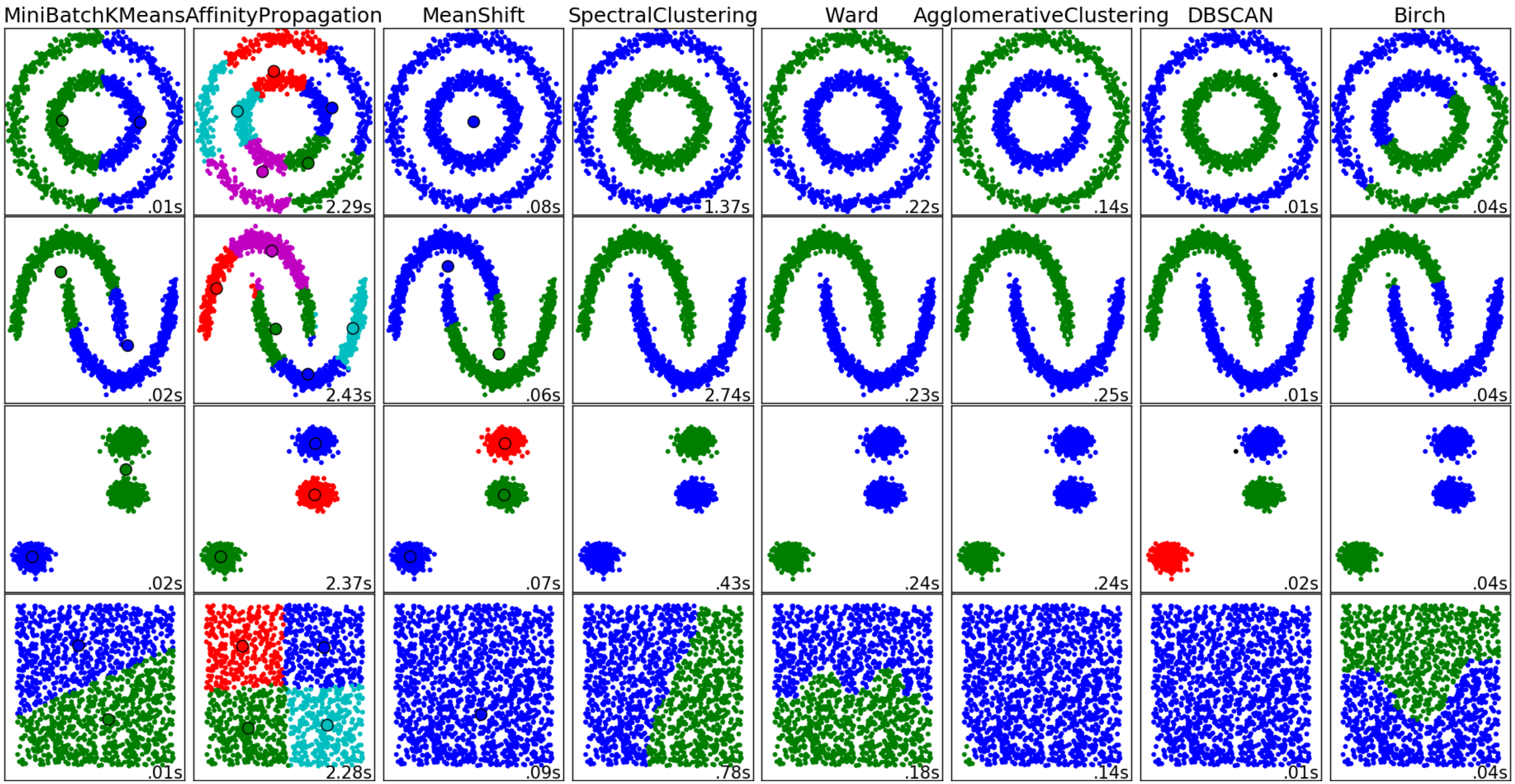


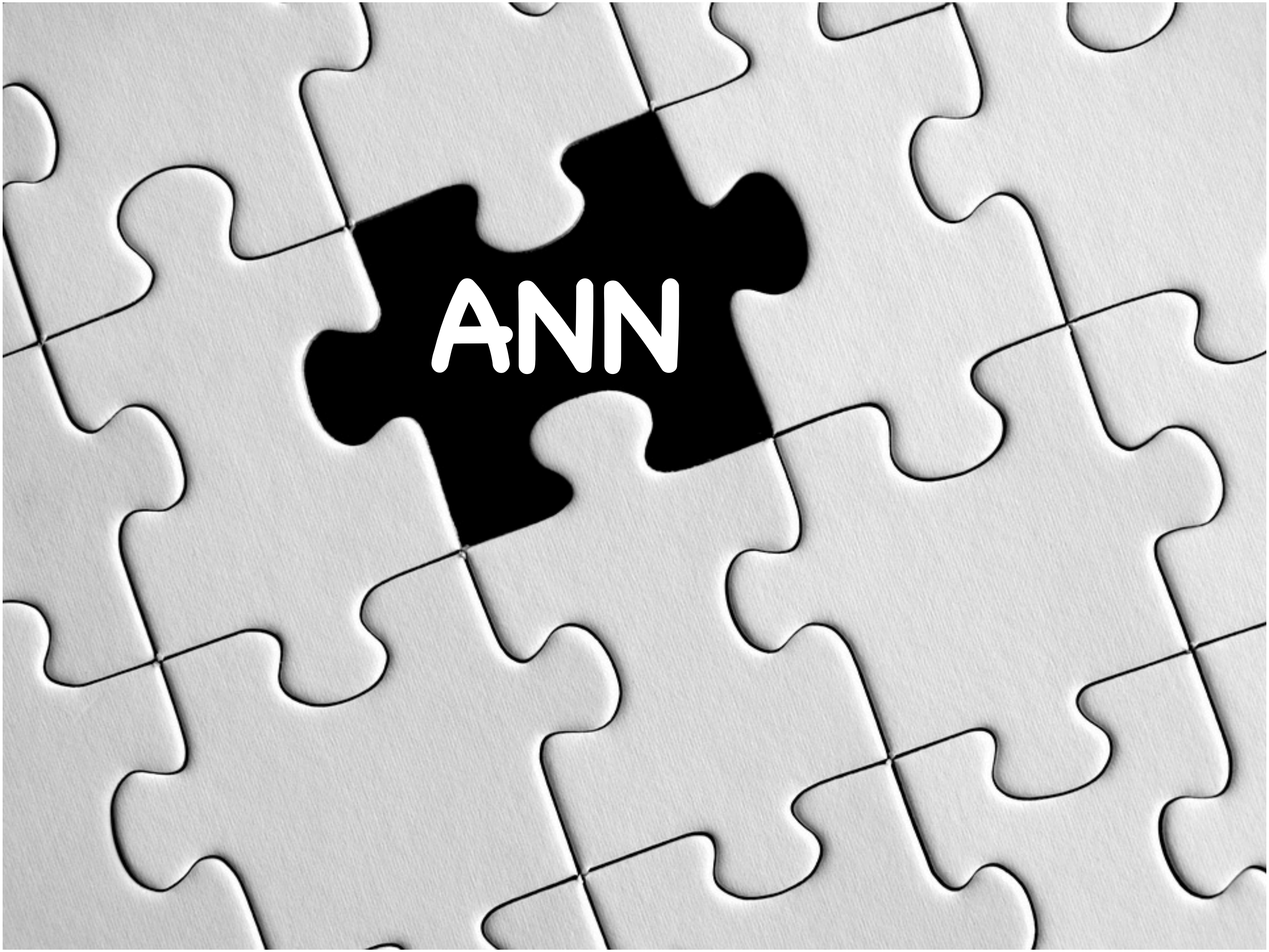
Regression

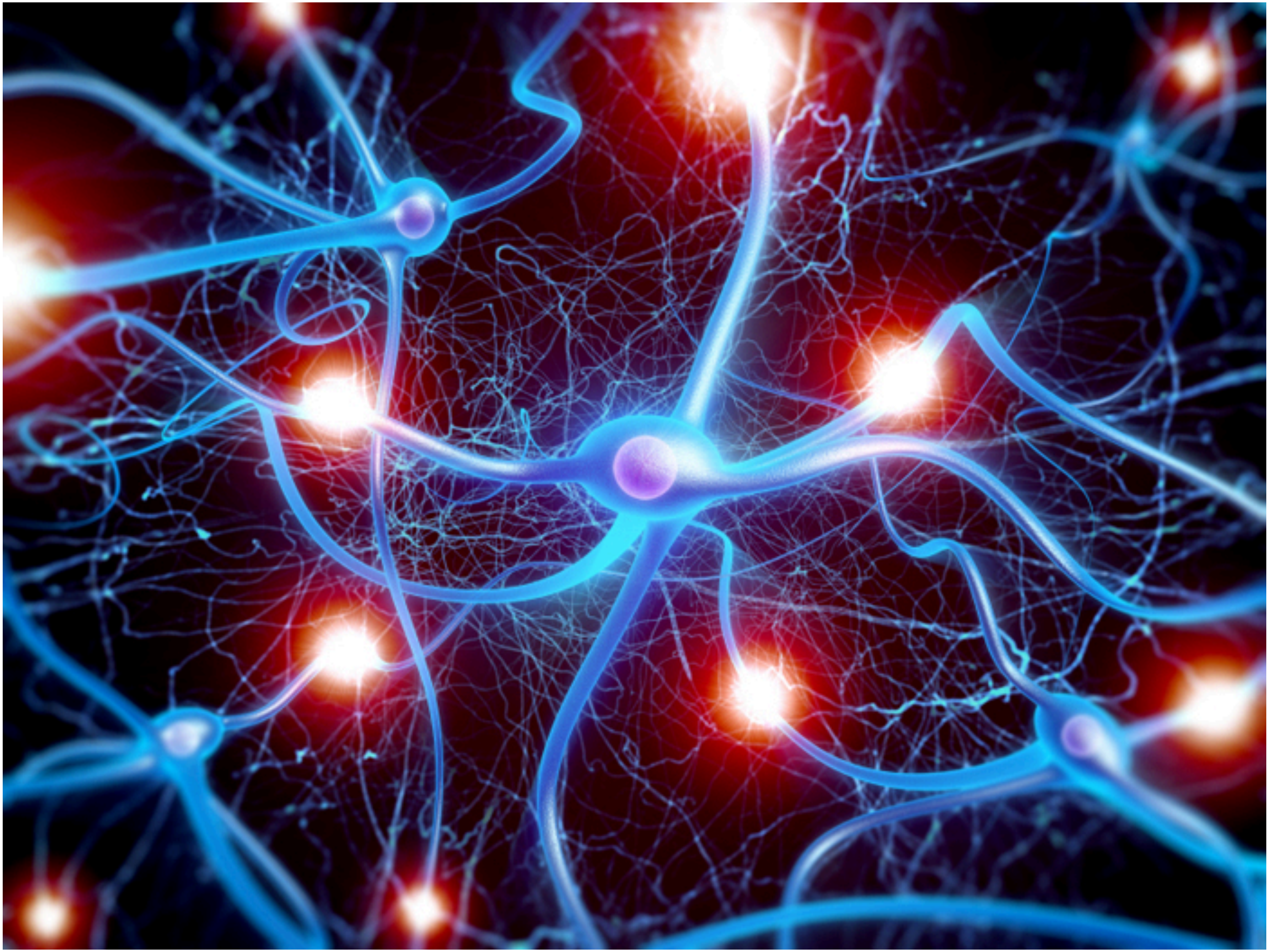
y
dependent variable
minimize errors
x
independent variable

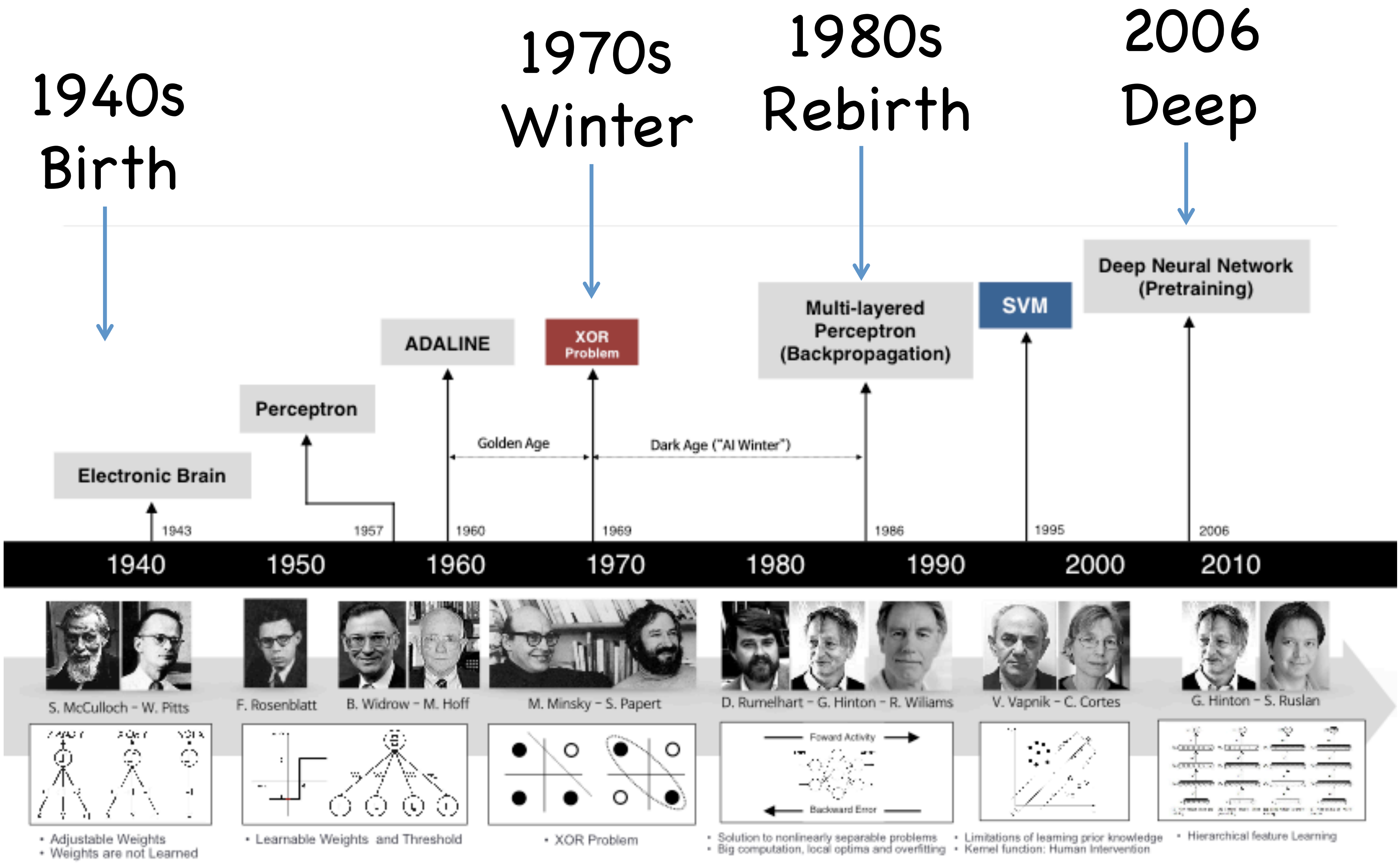

Classification

Clustering

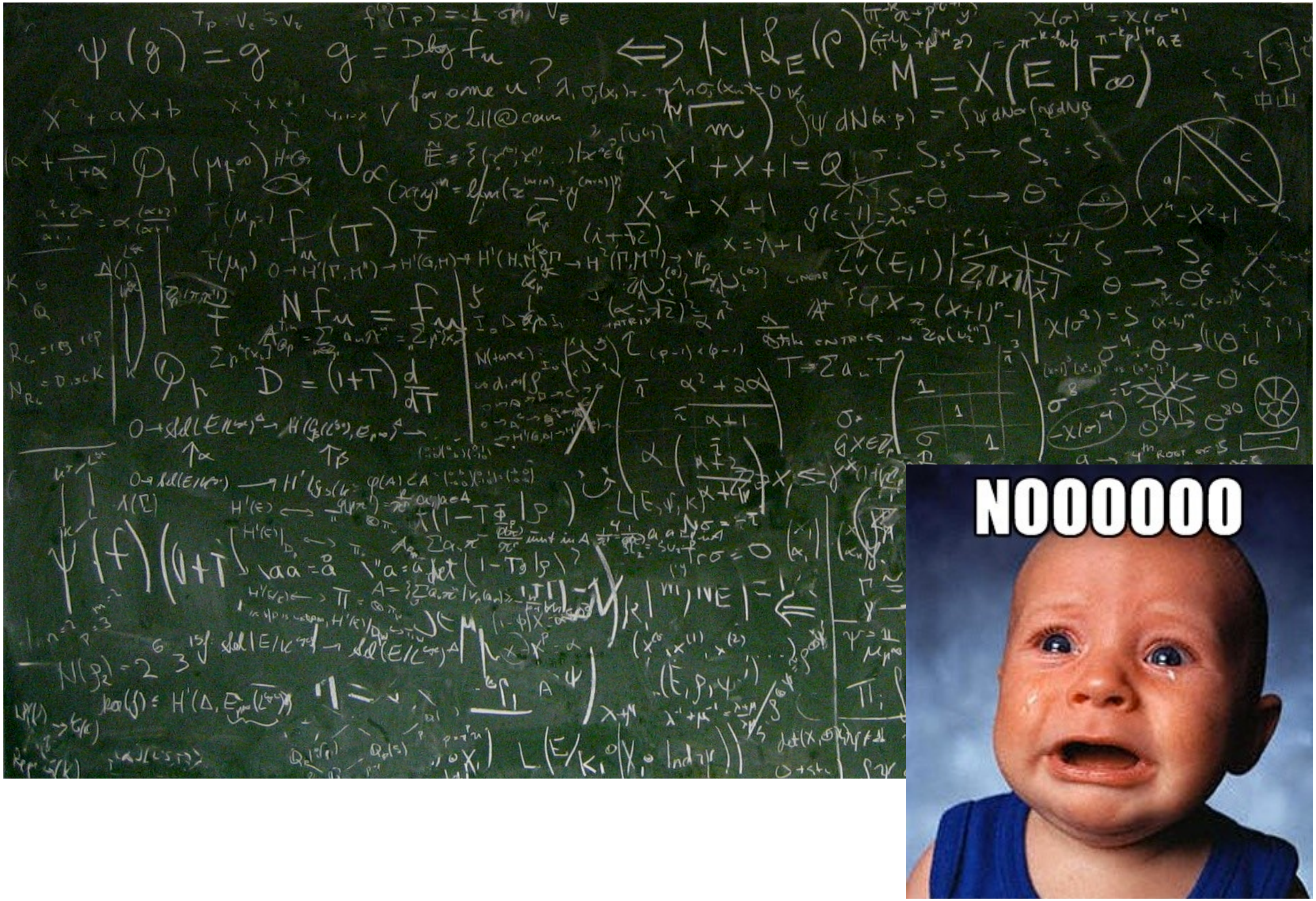

MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
.01s
2.29s
.08s
1.37s
.22s
.14s
.01s
.04s
.02s
2.43s
.06s
2.74s
.23s
.25s
.01s
.04s
.02s
2.37s
.07s
.43s
.24s
.24s
.02s
.04s
.01s
2.28s
.09s
.78s
.18s
.14s
.01s
.04s

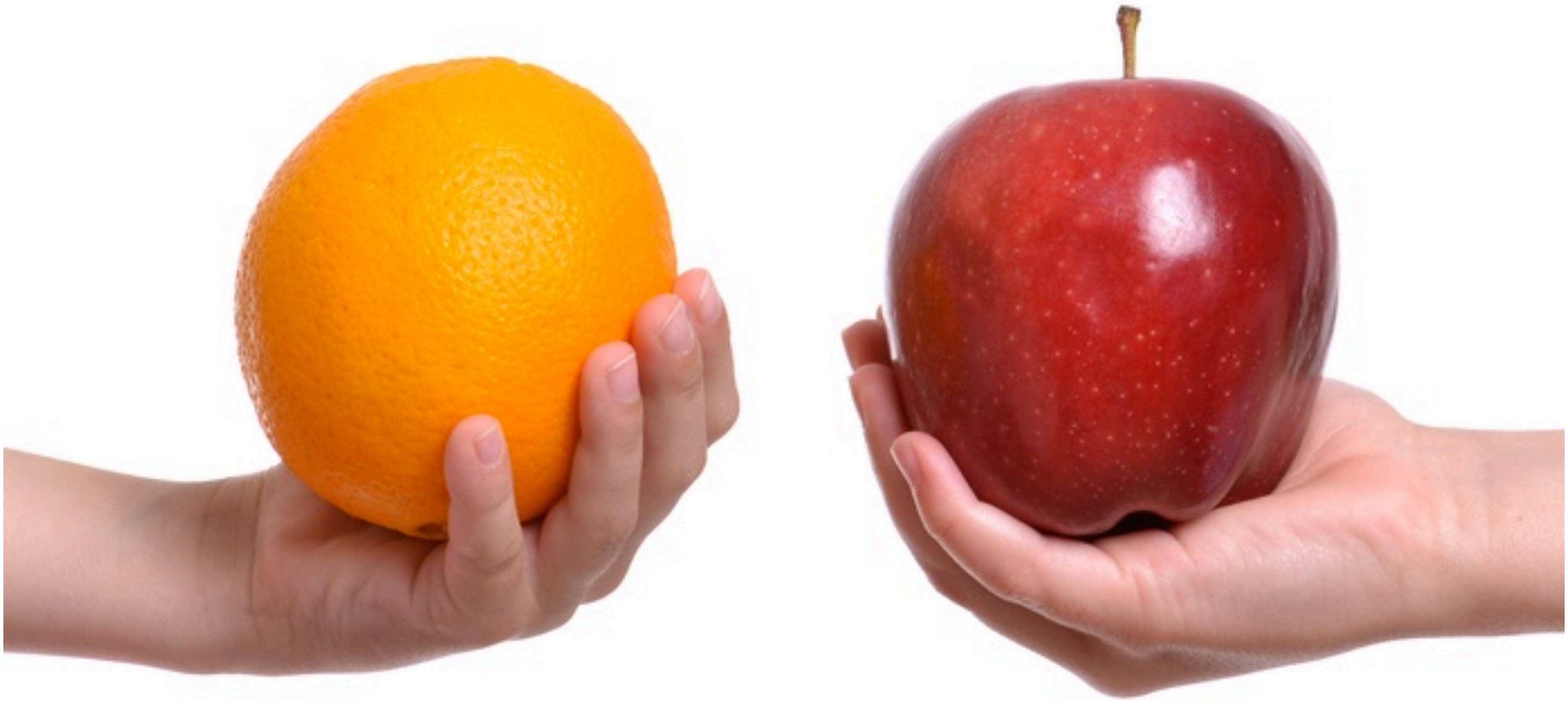

ANN



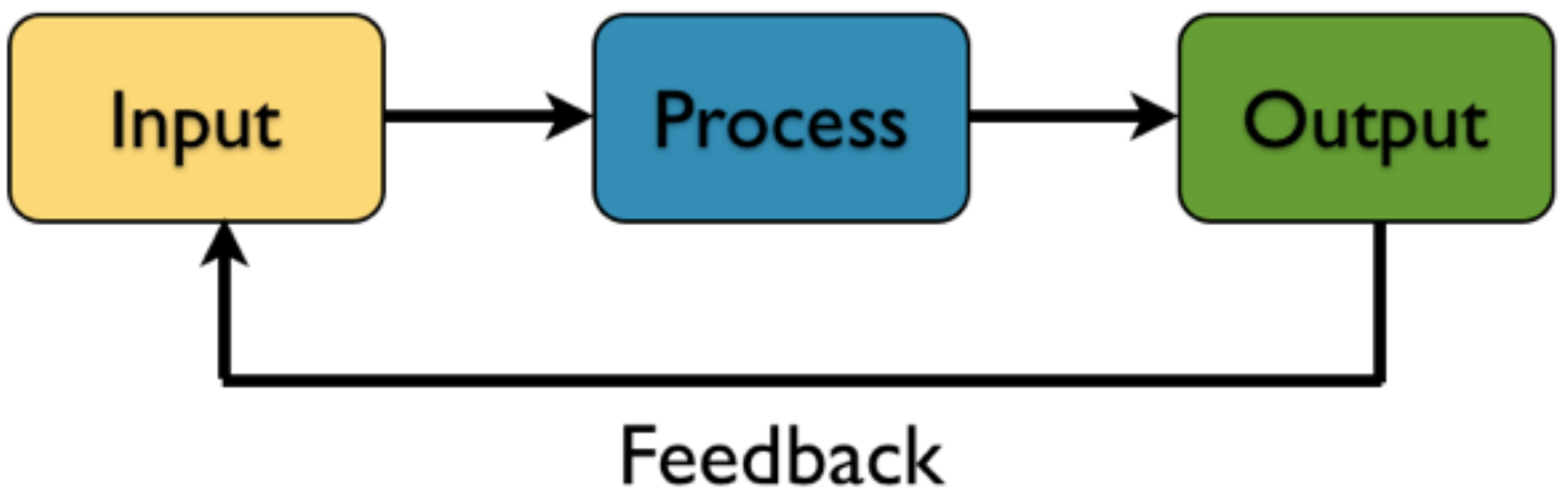


1940s
Birth
1970s
Winter
1980s
Rebirth
2006
Deep
Electronic Brain
Perceptron
ADALINE
XOR
Problem
Multi-layered
Perceptron
(Backpropagation)
SVM
Deep Neural Network
(Pretraining)
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
Forward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

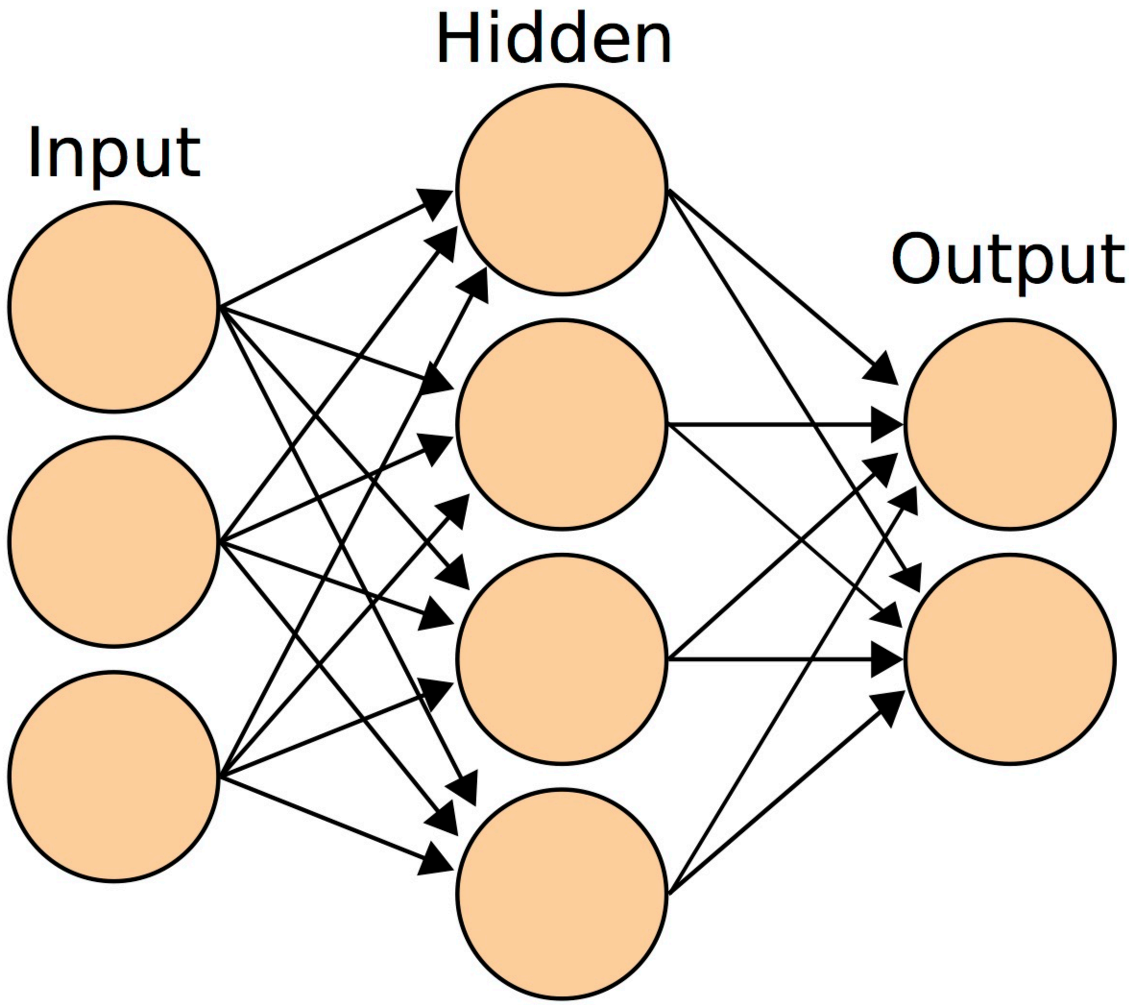

NOOOOOO



ANN in simple view



ANN jargons

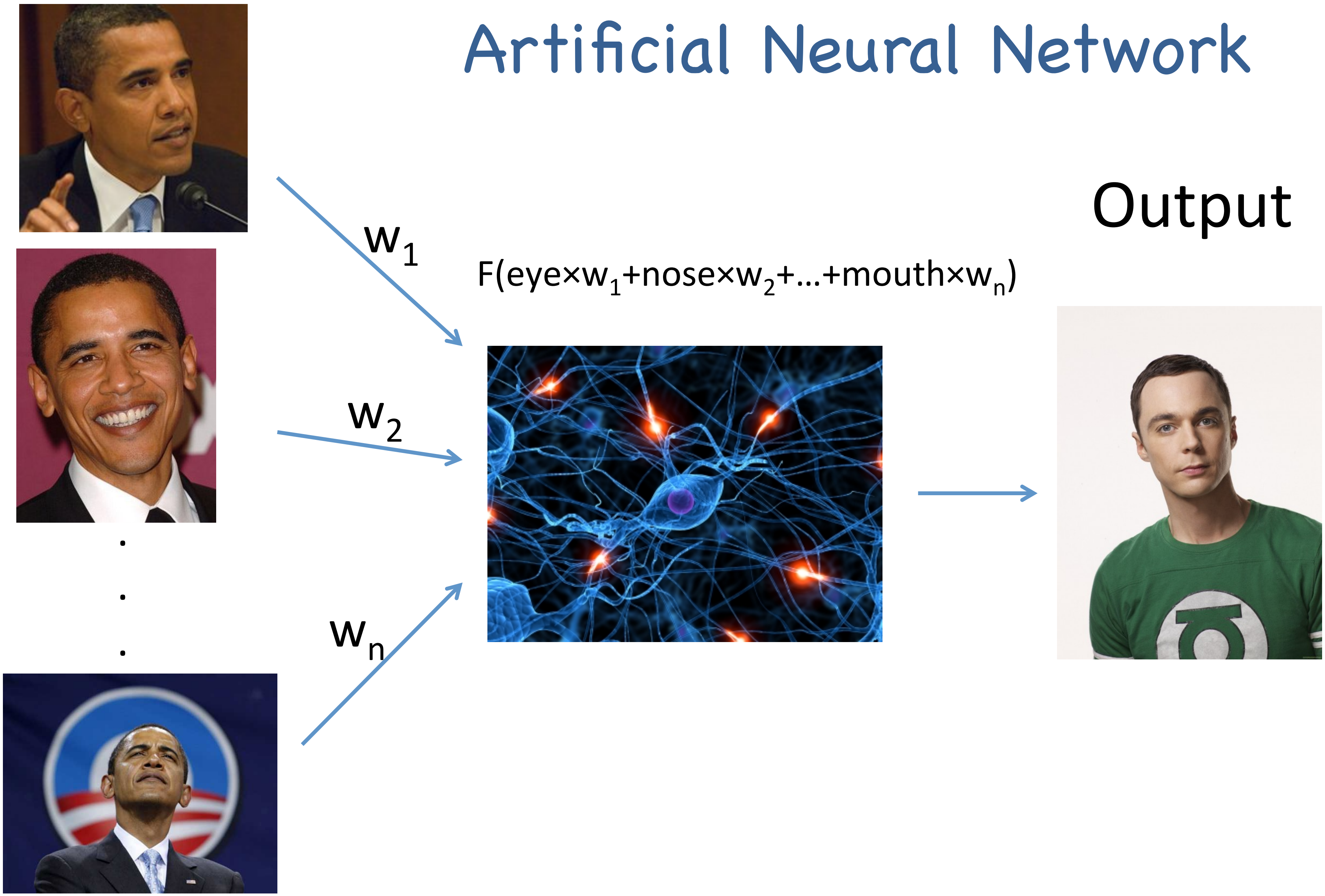


What're the weights

YOU'RE IN MY SPOT

Intuitive
Artificial Neural Network

Input



Intuitive
Artificial Neural Network

Input

.
.
.

w_1

w_2

w_n

F(eye×w_1+nose×w_2+…+mouth×w_n)

Output

error
feedback

Intuitive
Artificial Neural Network

Input

.
.
.

w_1

w_2

w_n

$F(eye\times w_1+nose\times w_2+\ldots+mouth\times w_n)$

Output

error feedback

Intuitive
Artificial Neural Network

Input

.
.
.

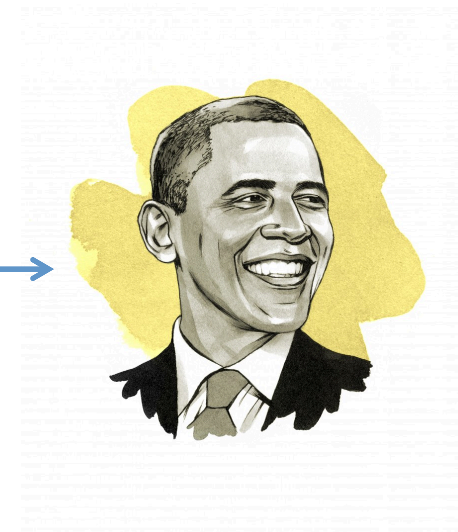
w_1

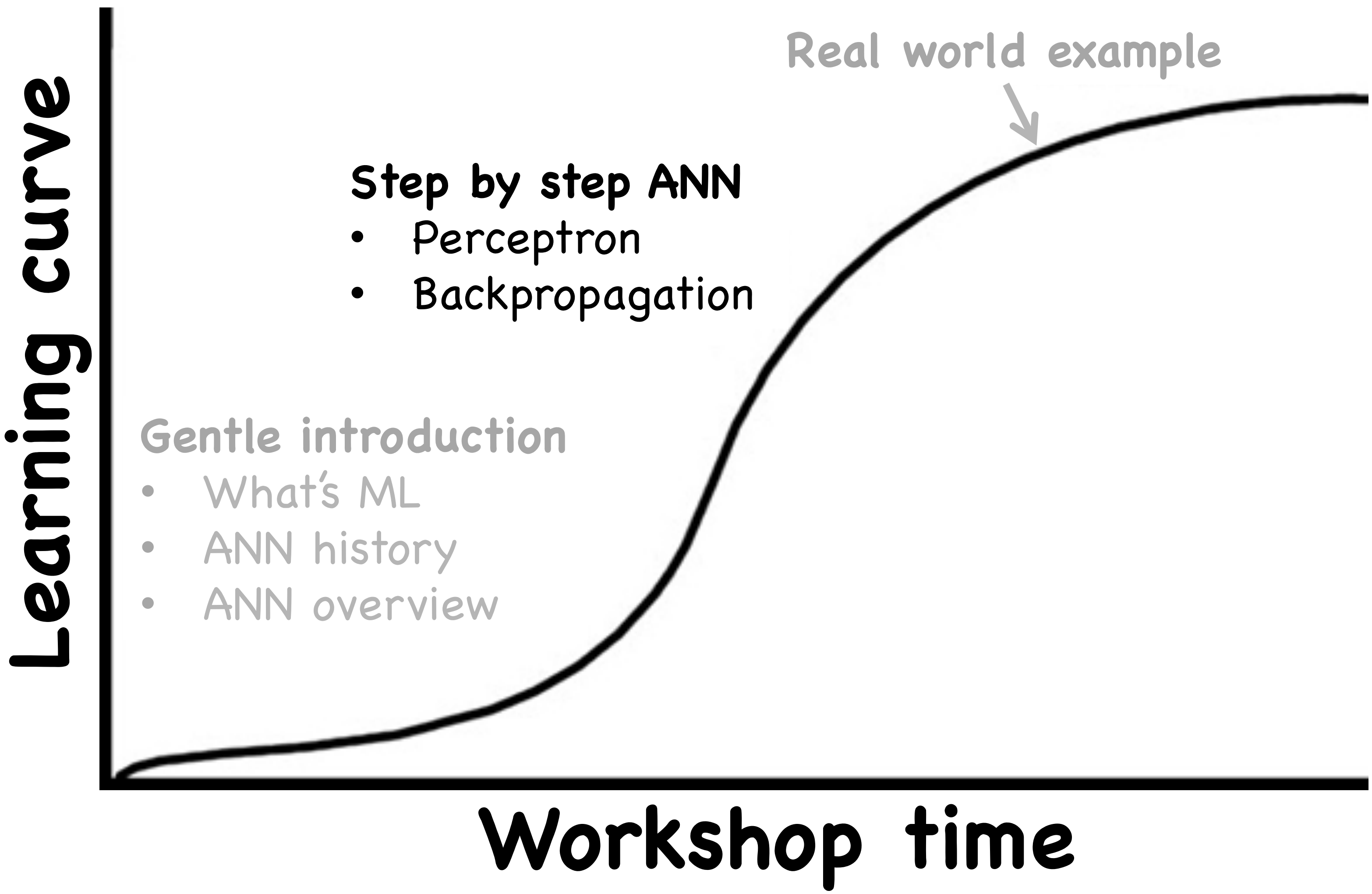
w_2

w_n

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

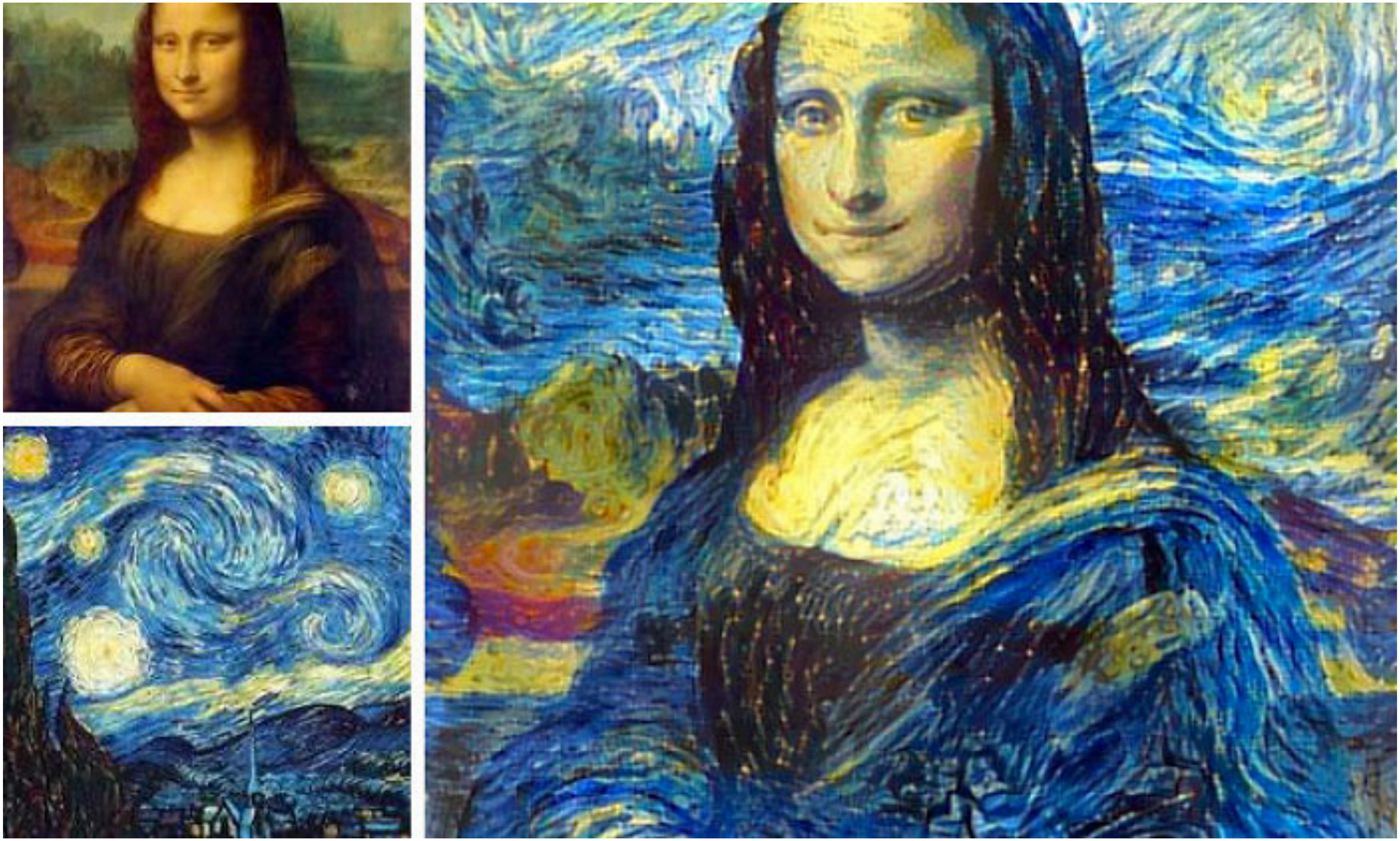
Output



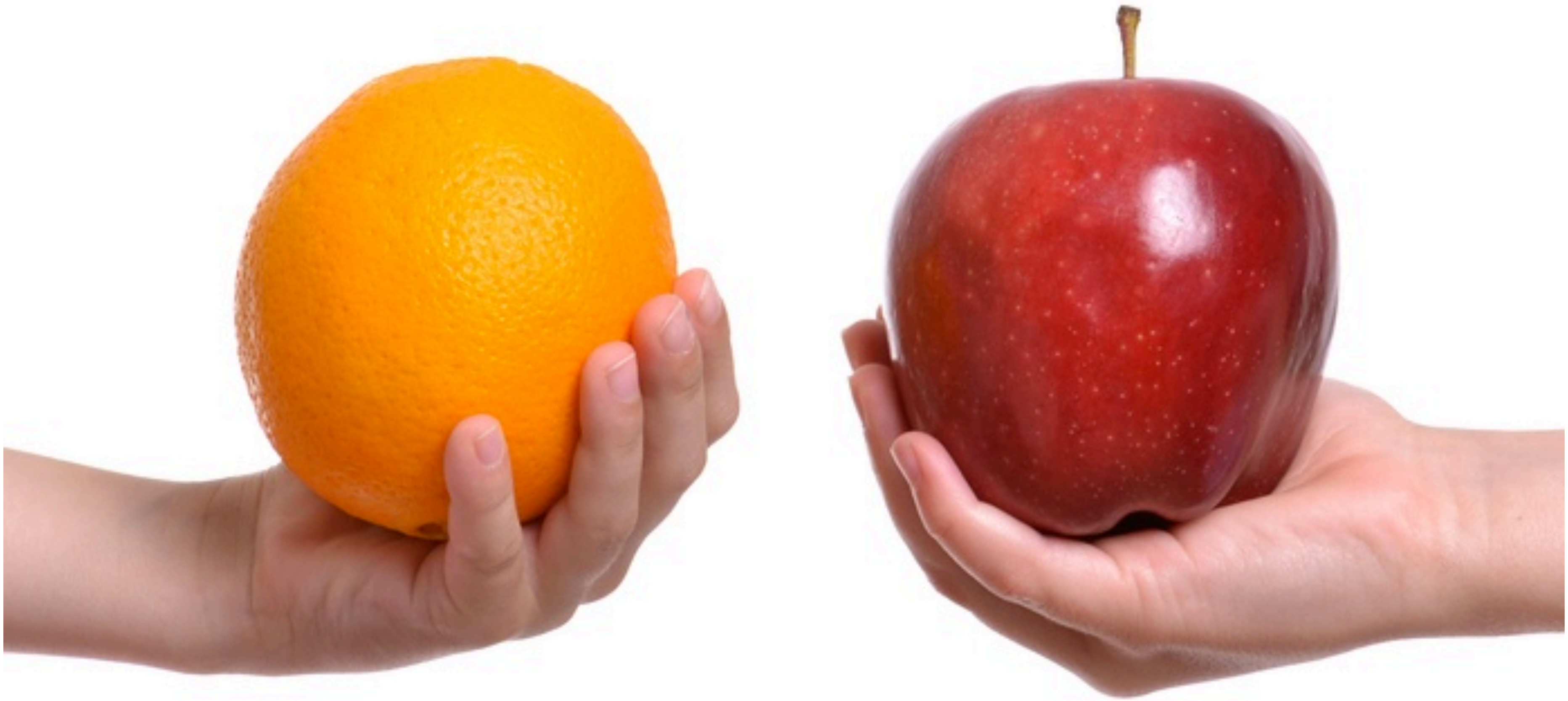

Real world example
Step by step ANN
• Perceptron
• Backpropagation
Gentle introduction
• What's ML
• ANN history
• ANN overview
Learning curve
Workshop time

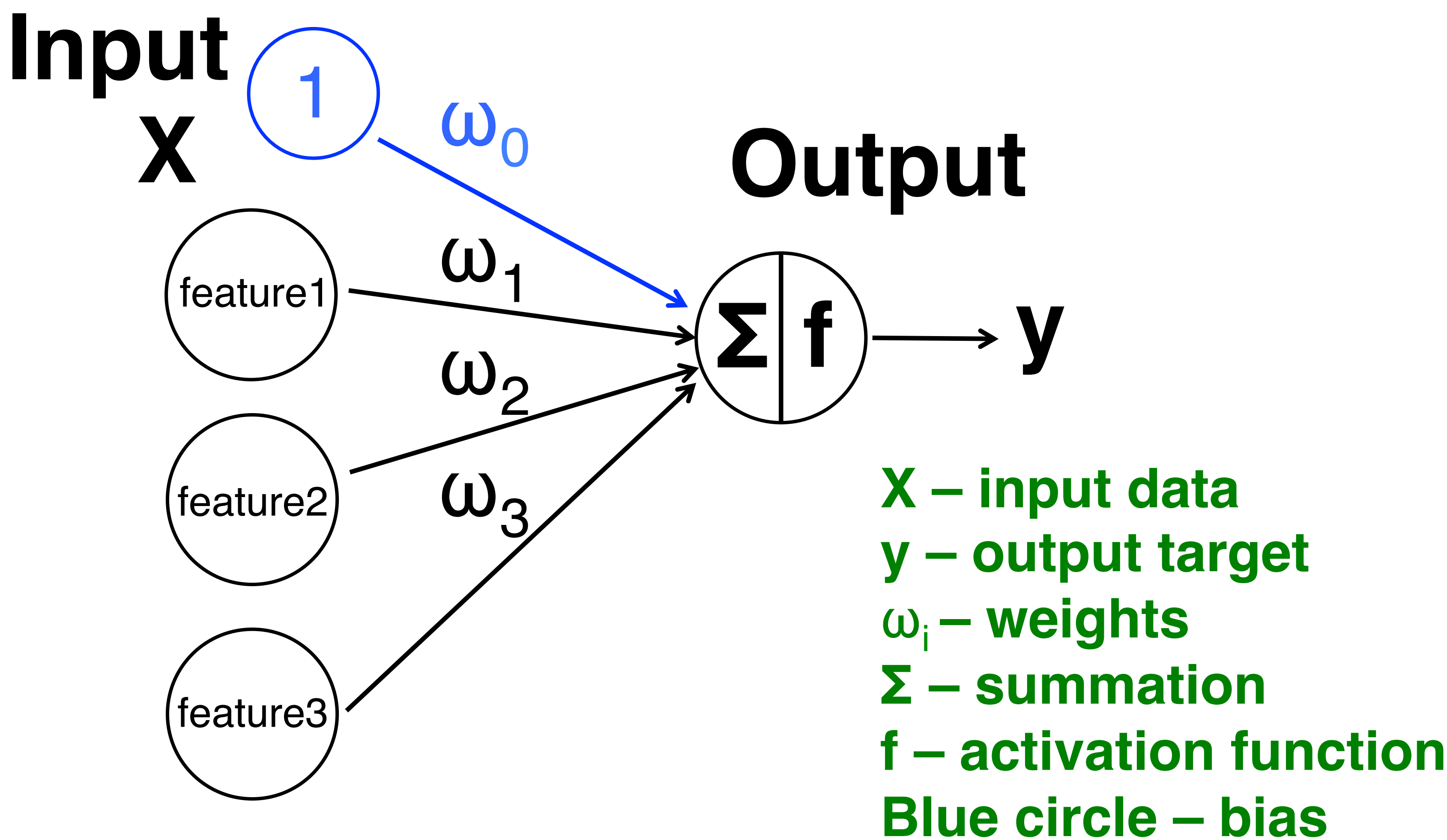
Application: Learn arts

https://arxiv.org/abs/1508.06576v1
https://deepart.io/



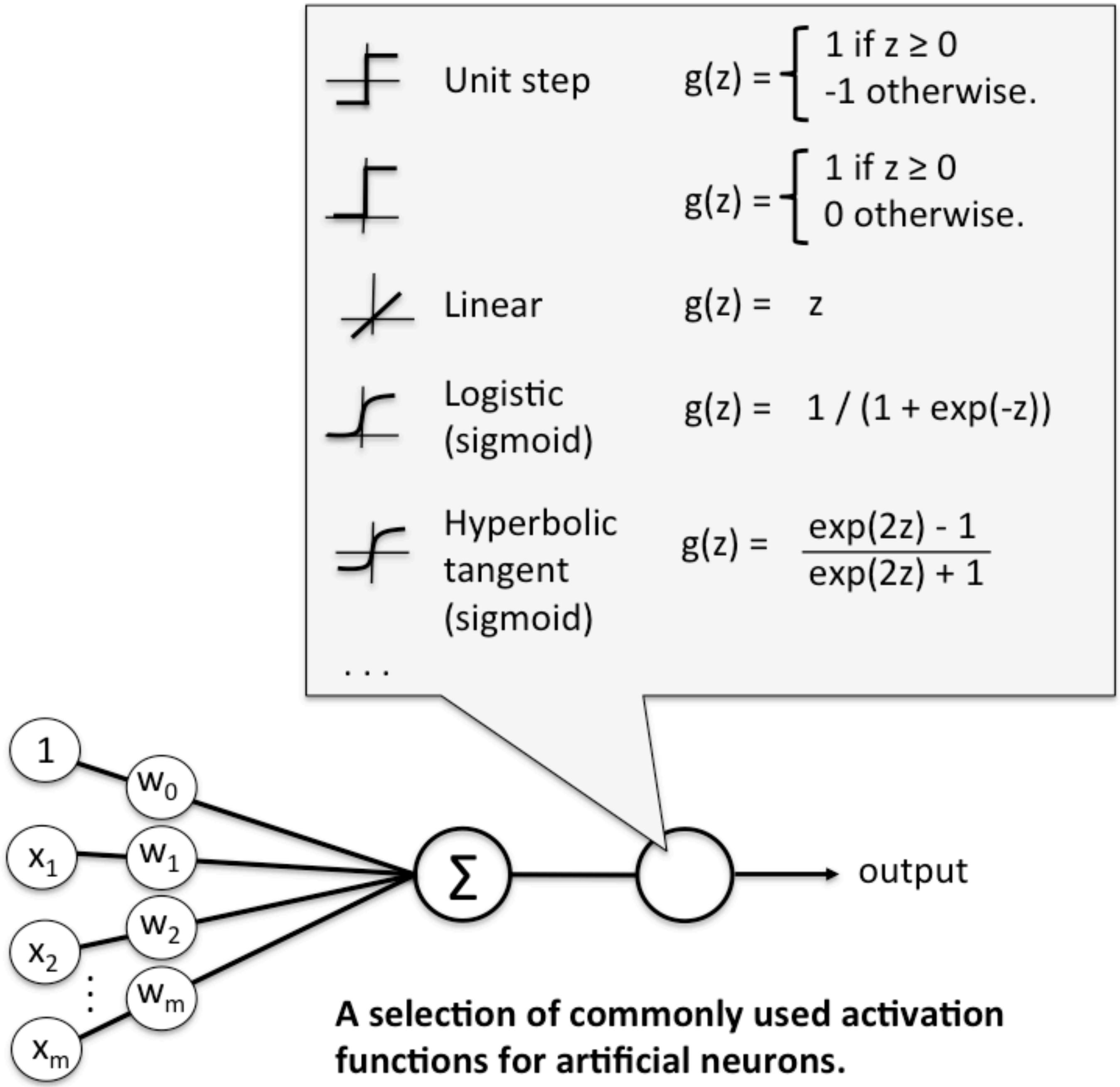
http://junkhost.com/2016/03/man-combines-random-peoples-photos-using-neural-networks-and-the-results-are-amazing/





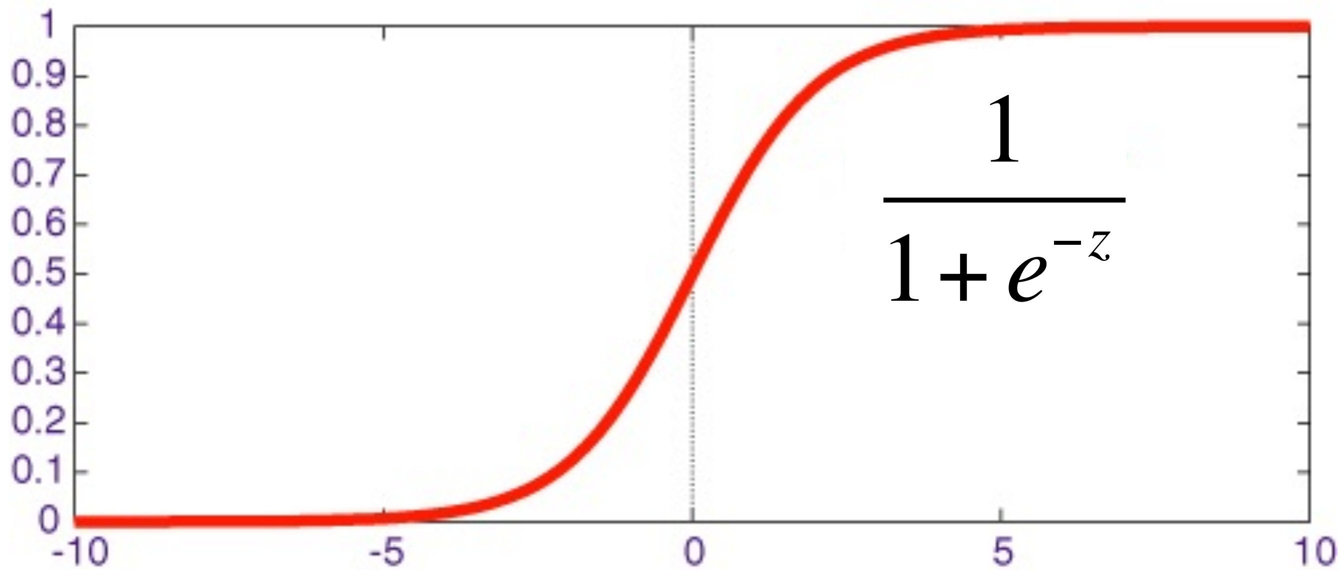
$$\Sigma = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \ldots + \omega_n x_n$$

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



A selection of commonly used activation functions for artificial neurons.

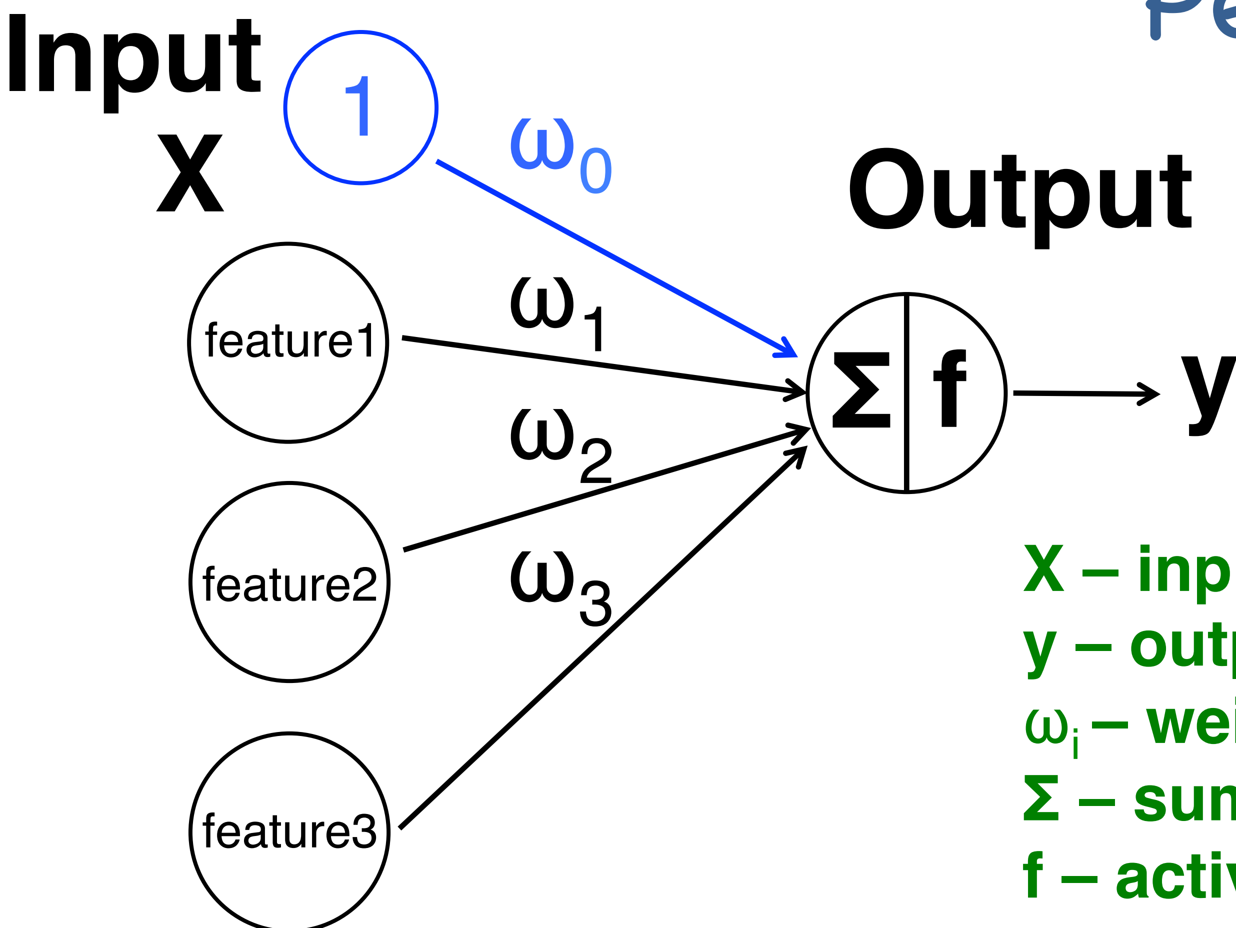
More on activation function



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

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

Perceptron



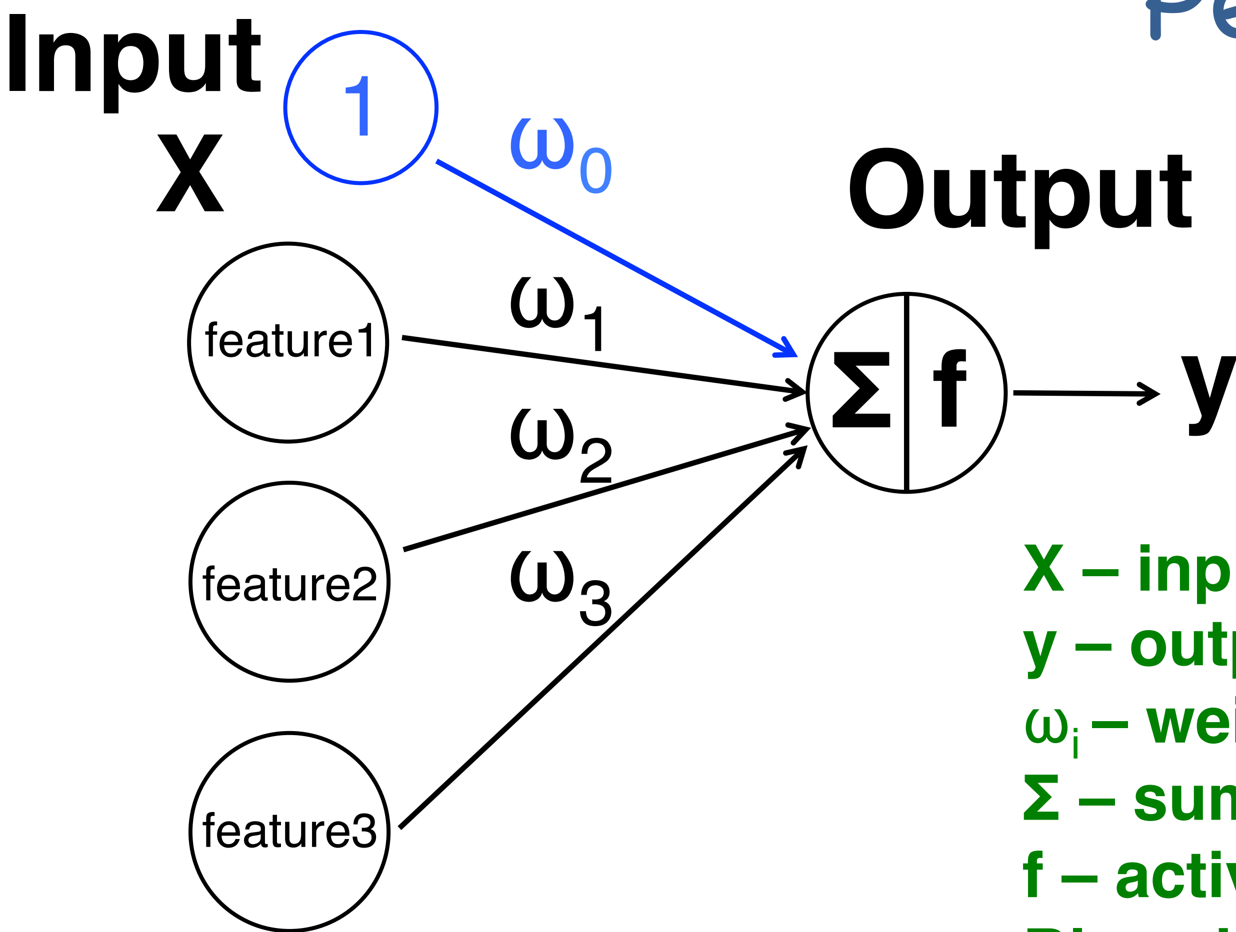
X – input data
y – output target
ω_i – weights
Σ – summation
f – activation function
Blue circle – bias

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

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

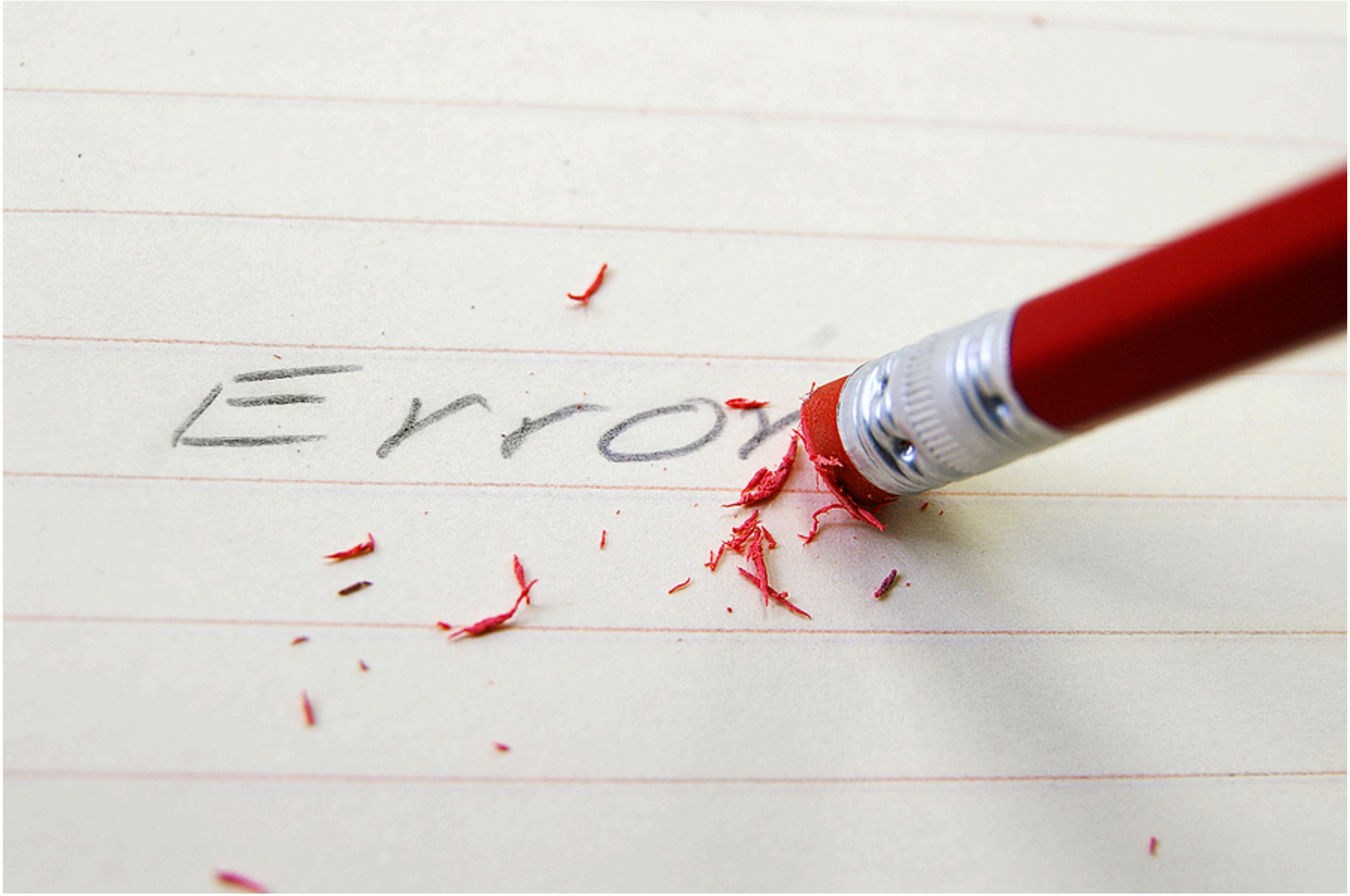
y **This is our estimation**

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:

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


HOW?

Weights update rules

Weights Delta = Error × slope × input

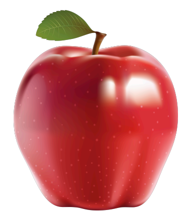
How much we will update
the weights for next time

Look at errors closer

Error = Target - Estimation

Target

0 **1**



Look at errors closer

Error = Target - Estimation

Target

 **0** **1** 

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**

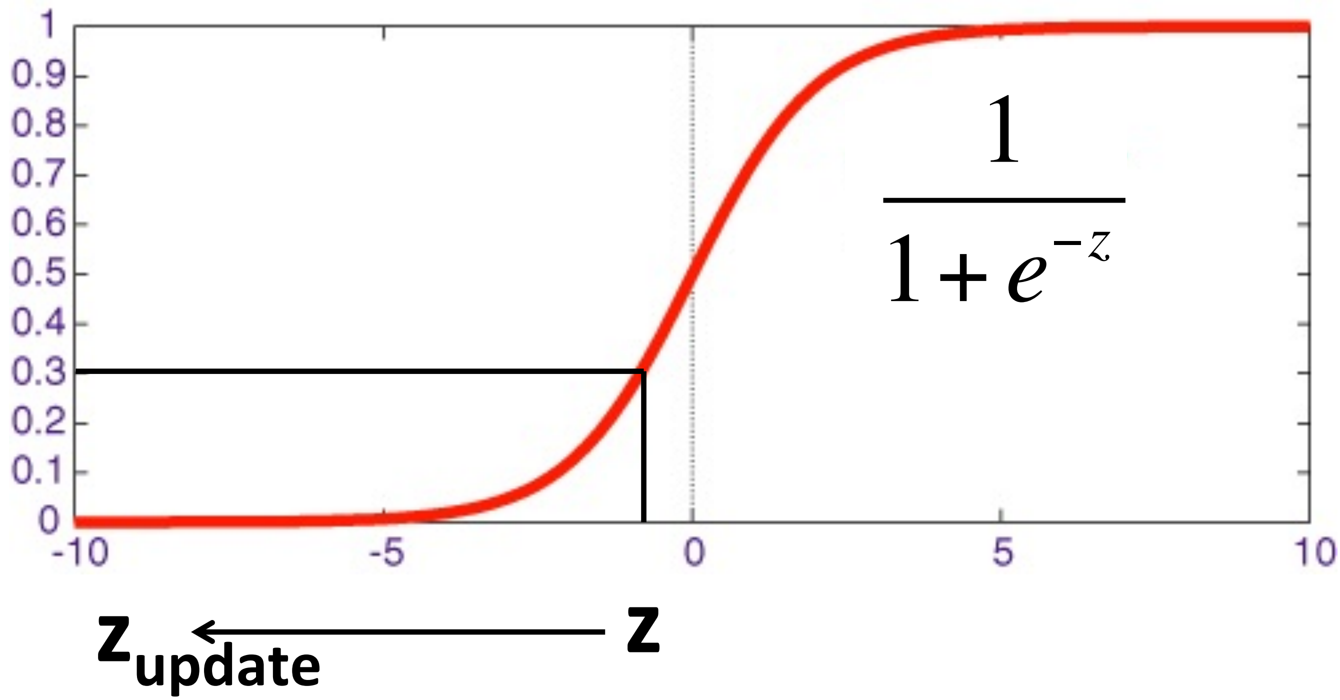
Error = 0 – 0.3 =-0.3

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

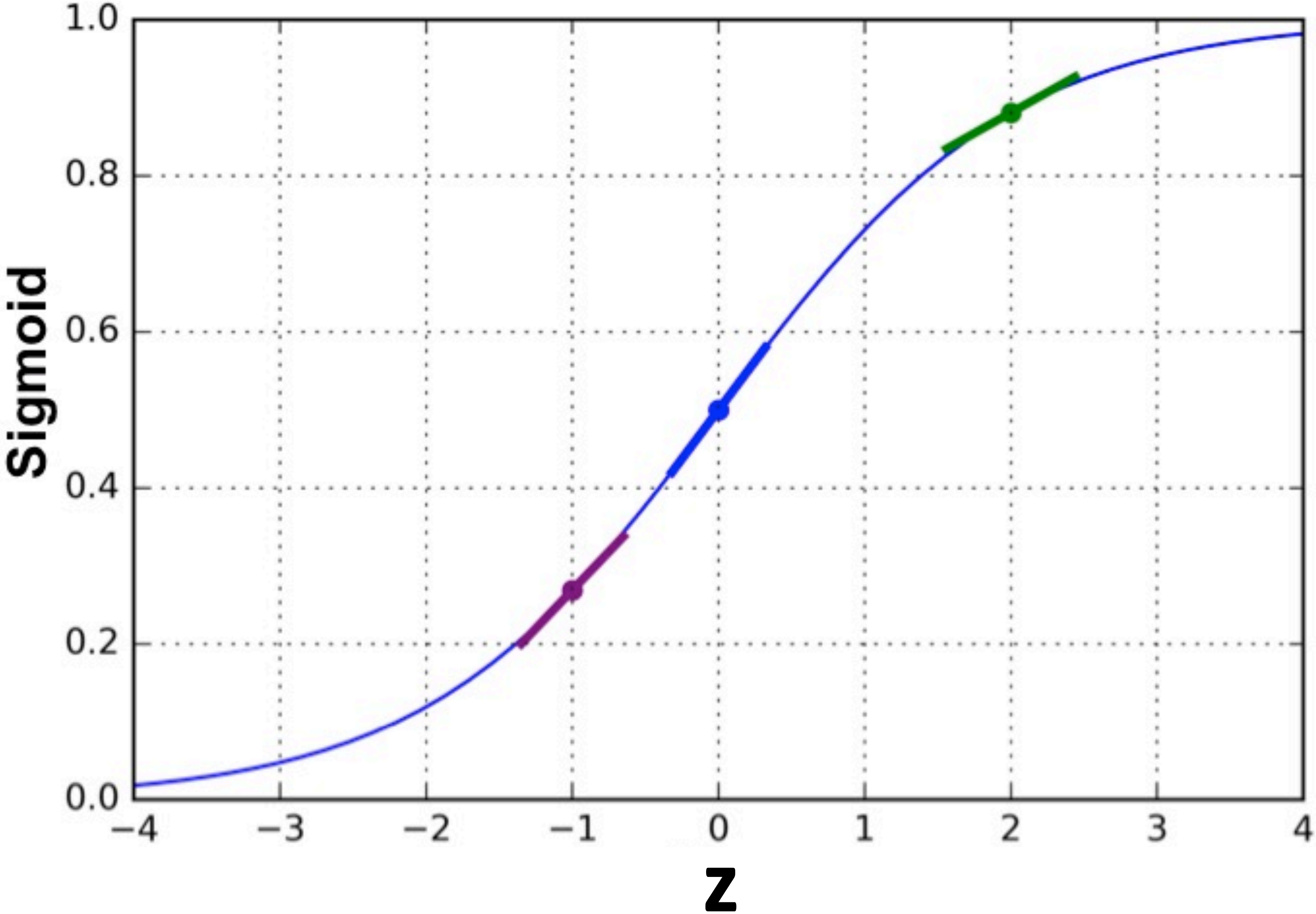
$$z = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3$$

We need reduce weights!

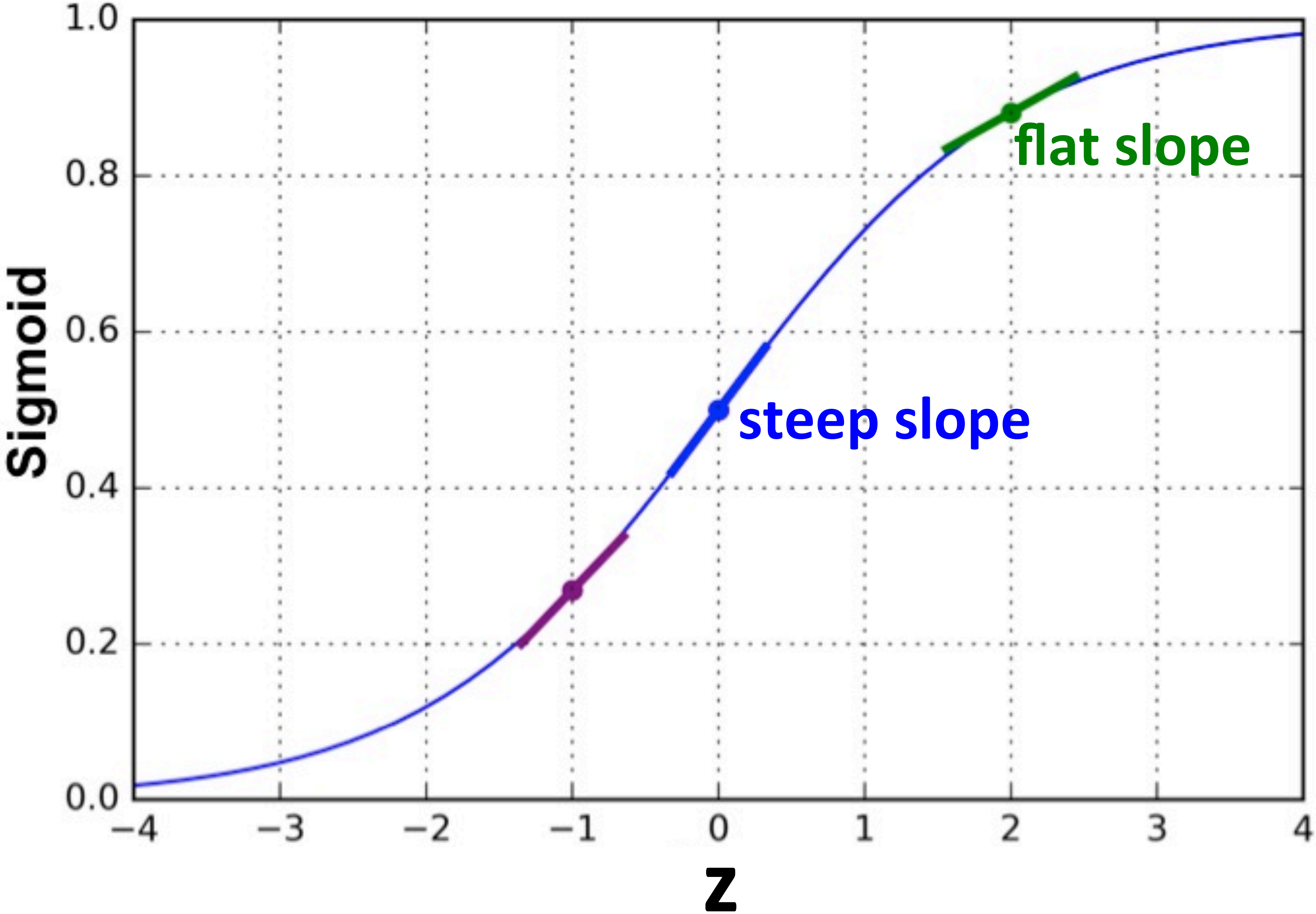
But what if the inputs are negative

Weights update rules

Weights Delta = Error × input



Sigmoid
1.0
0.8
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
1
2
3
4
z



1.0
0.8
0.6
0.4
0.2
0.0
Sigmoid
−4
−3
−2
−1
0
1
2
3
4
z
flat slope
steep slope

Weights update rules

Weights Delta = Error × slope × input

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

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

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

View from Gradient Descent

$$\frac{\partial E}{\partial \omega_{ik}} = \frac{\partial}{\partial \omega_{ik}}\left(\frac{1}{2}\sum_{k=1}^{N}(y_k - t_k)^2\right)$$

$$= \frac{\partial}{\partial \omega_{ik}}\left(\frac{1}{2}\sum_{k=1}^{N}[f(\sum_{i=0}^{L}\omega_{ik}x_i) - t_k]^2\right)$$

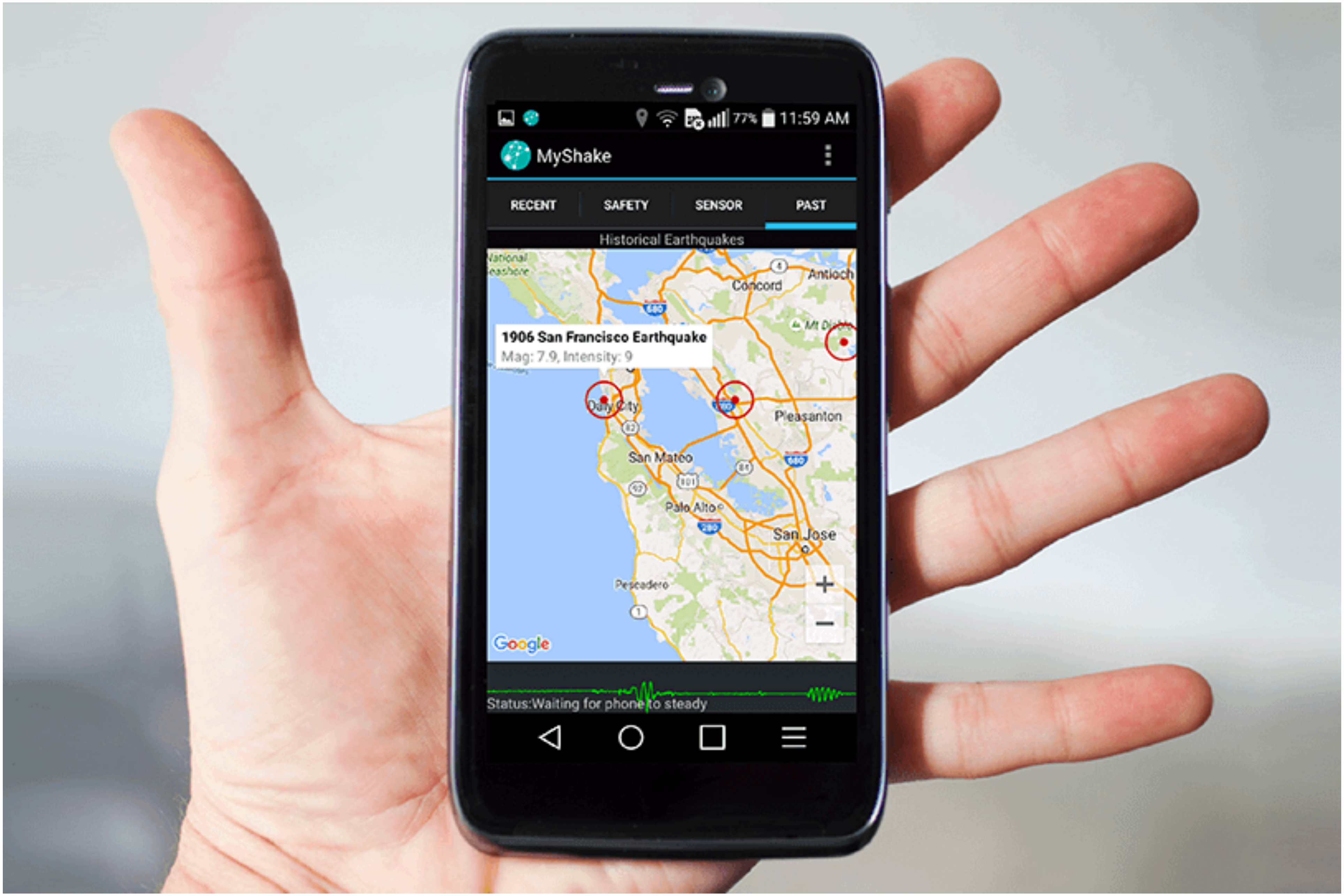
$$= \frac{1}{2}\sum_{k=1}^{N}2(y_k - t_k)\frac{\partial}{\partial \omega_{ik}}\left(f(\sum_{i=0}^{L}\omega_{ik}x_i) - t_k\right)$$

$$= \sum_{k=1}^{N}(y_k - t_k)\frac{\partial f}{\partial \omega_{ik}}x_i =$$ **Error × slope × input**

Weights update rules

Weights Delta = Error × slope × input


Learn from example

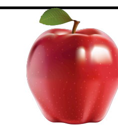

11:59 AM
MyShake
RECENT
SAFETY
SENSOR
PAST
Historical Earthquakes
1906 San Francisco Earthquake
Mag: 7.9, Intensity: 9
Concord
Antioch
Pleasanton
San Mateo
Palo Alto
San Jose
Pescadero
Google
Status:Waiting for phone to steady

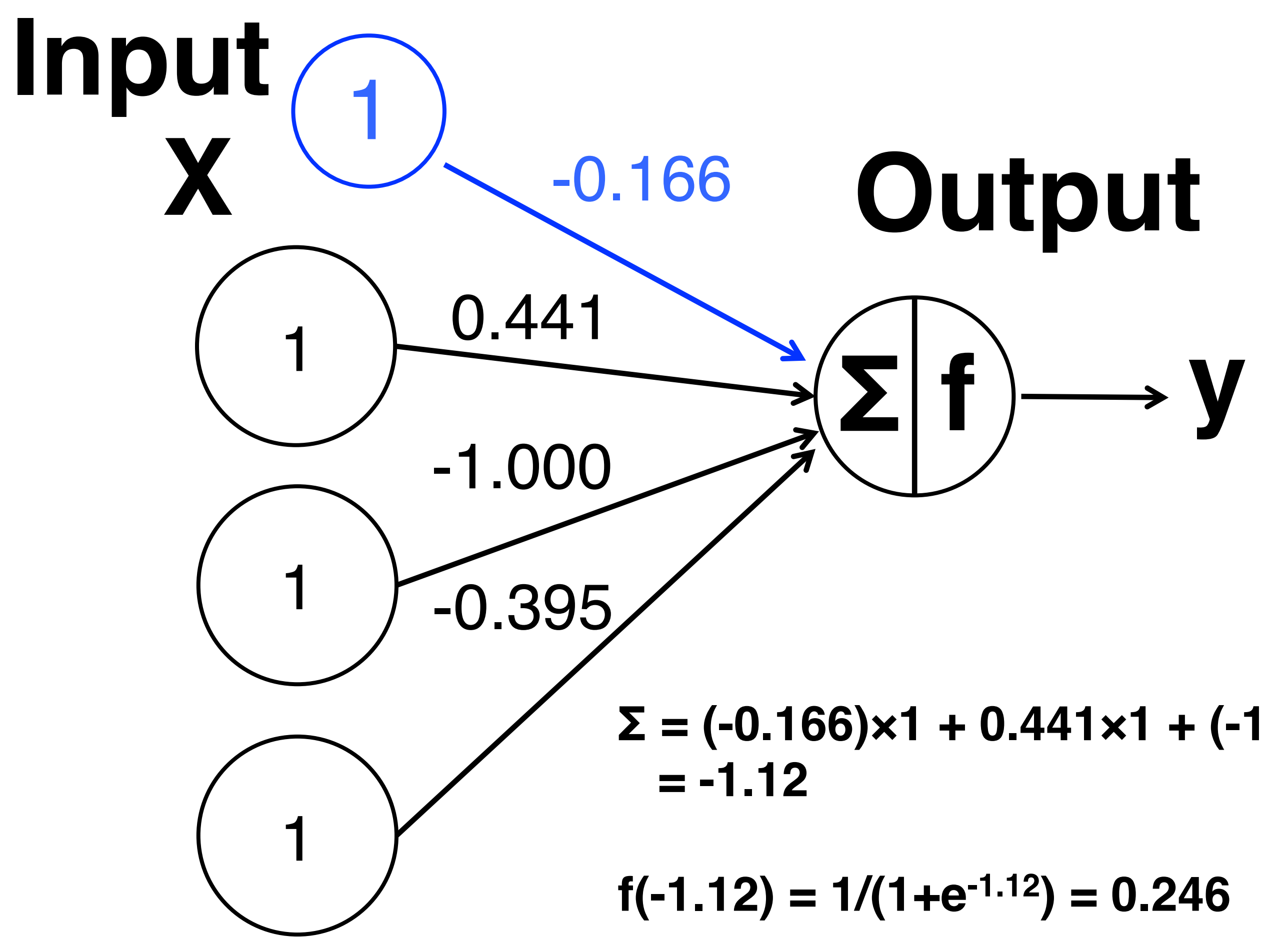
Learn from Example

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

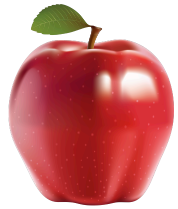




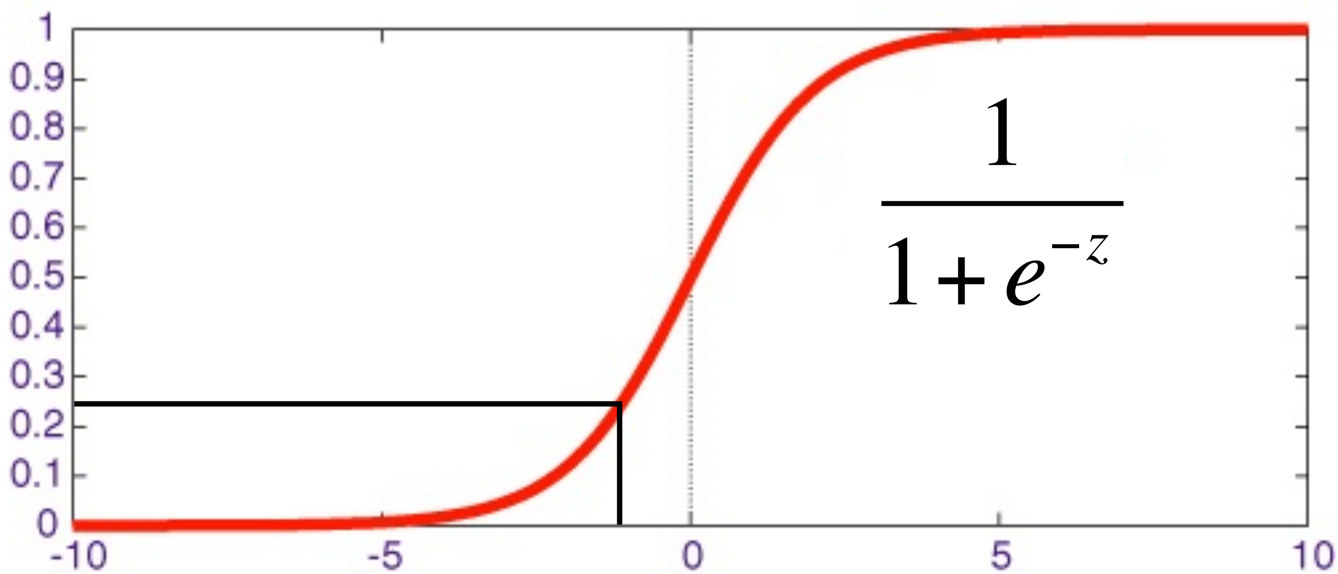
$$\Sigma = (-0.166)\times 1 + 0.441\times 1 + (-1)\times 1 + (-0.395)\times 1$$
$$= -1.12$$

$$f(-1.12) = 1/(1+e^{-1.12}) = 0.246$$

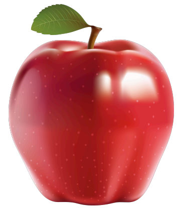
$$\text{Error} = 1 - 0.246 = 0.754$$

- **Target: 1** 
- **Estimation: 0.246**

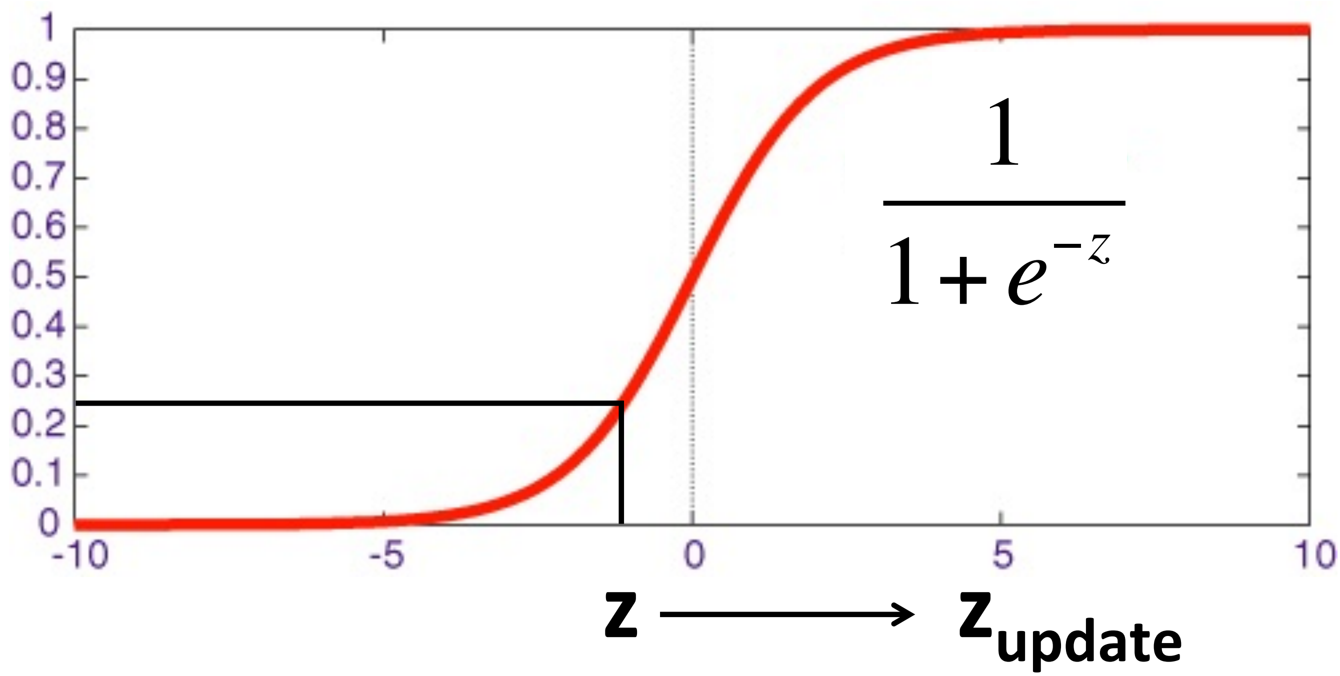
Error 0.754



$\mathbf{z} \longrightarrow \mathbf{z}_{\mathbf{update}}$

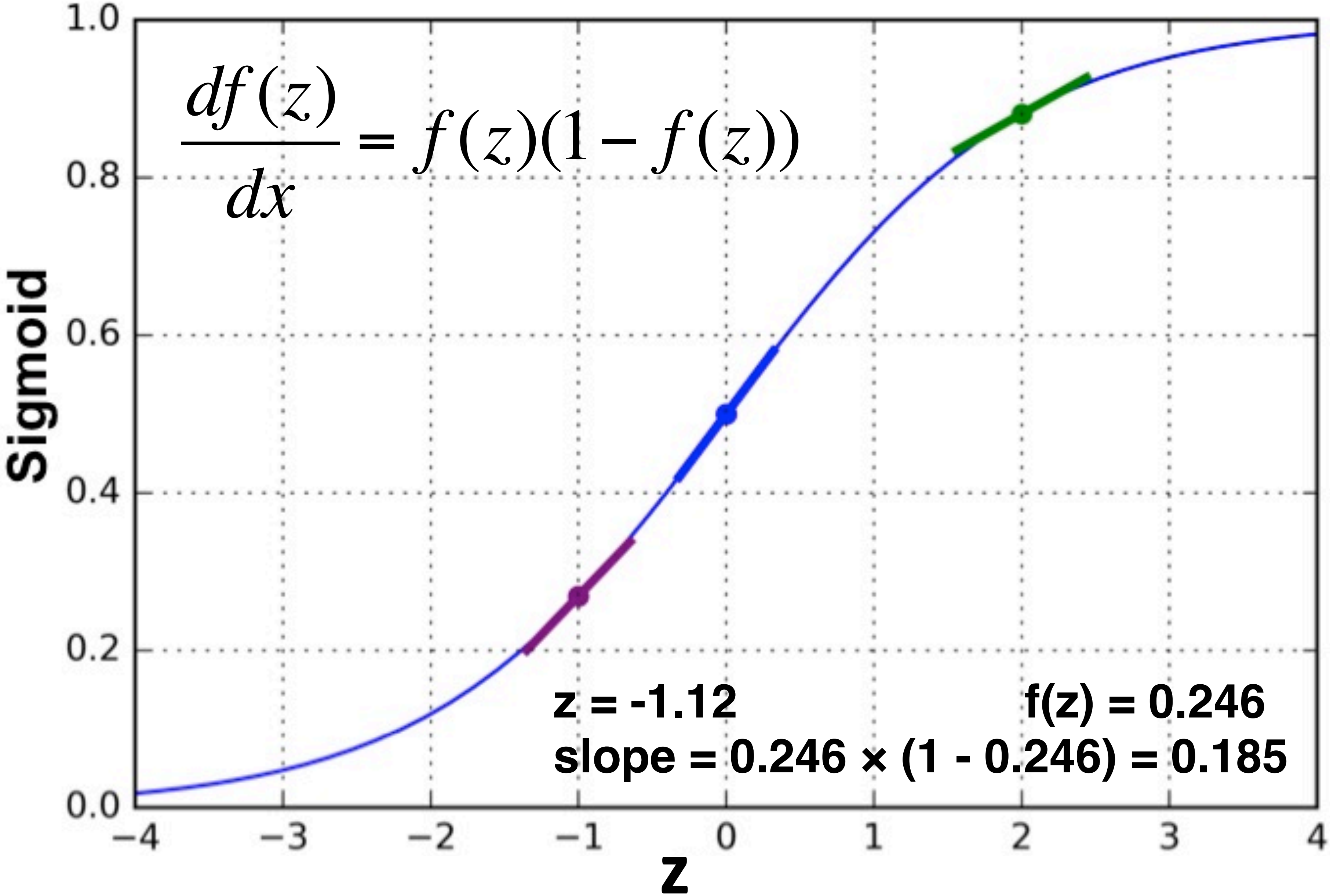
- **Target: 1** 
- **Estimation: 0.246**

Error 0.754



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

Weights delta = 0.754 × slope × input



$$\frac{df(z)}{dx} = f(z)(1 - f(z))$$
z = -1.12
f(z) = 0.246
slope = 0.246 × (1 - 0.246) = 0.185
Sigmoid
z
1.0
0.8
0.6
0.4
0.2
0.0
−4
−3
−2
−1
0
1
2
3
4

$$\text{Change item} = 0.754 \times 0.185 \times \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.139 \\ 0.139 \\ 0.139 \\ 0.139 \end{bmatrix}$$

Changes of the error

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

Original Weights

updates

Changes of the error

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

Error of next iteration

$z = (-0.027)\times1 + 0.580\times1 + (-0.861)\times1 + (-0.256)\times1 = -0.564$

$f(z) = 0.637$

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

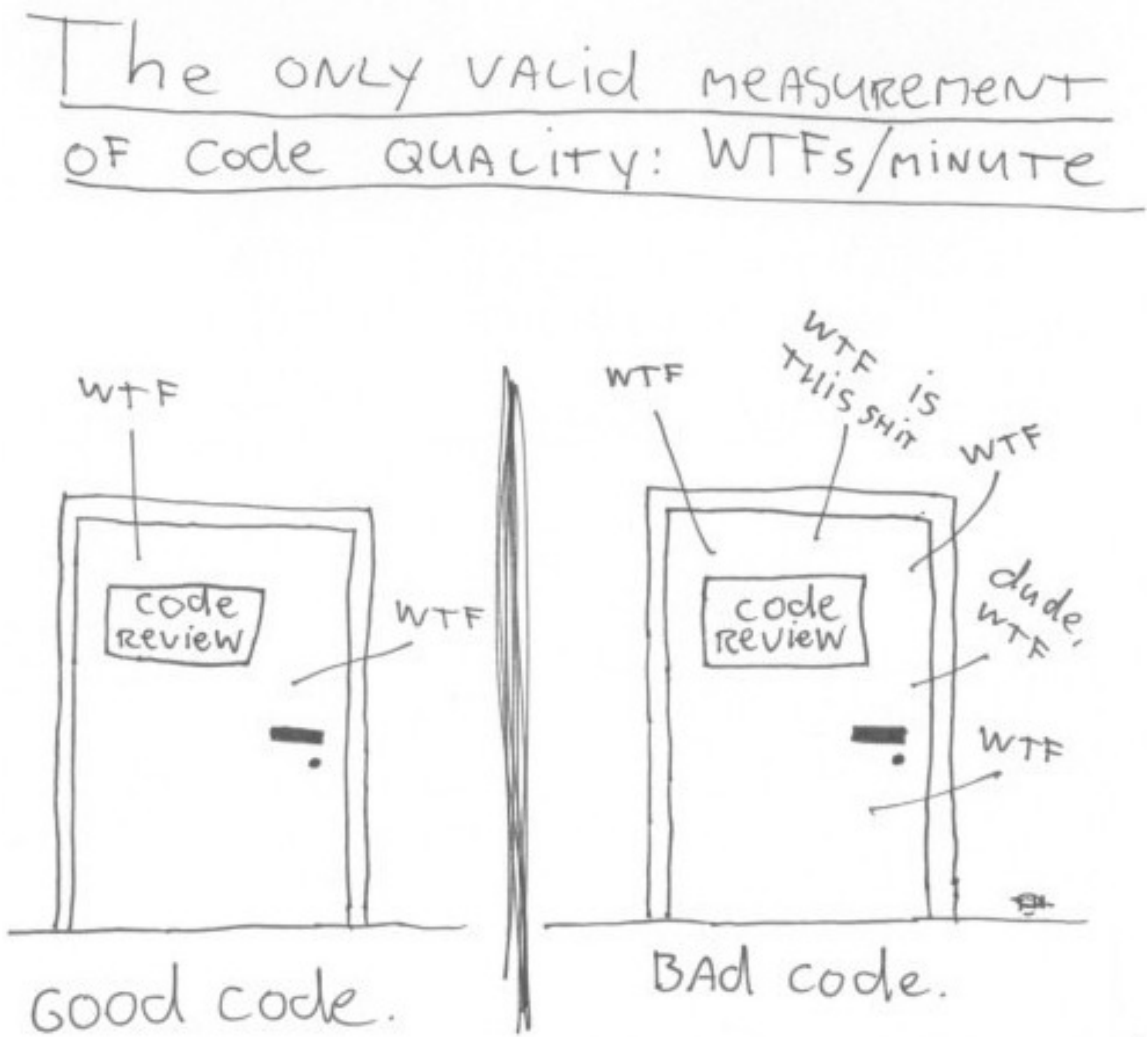
Changes of the error

0.754

0.363

Iterate many times

Go to notebook 01



Application: DeepDrumpf

https://twitter.com/DeepDrumpf

DeepDrumpf @DeepDrumpf · Mar 9

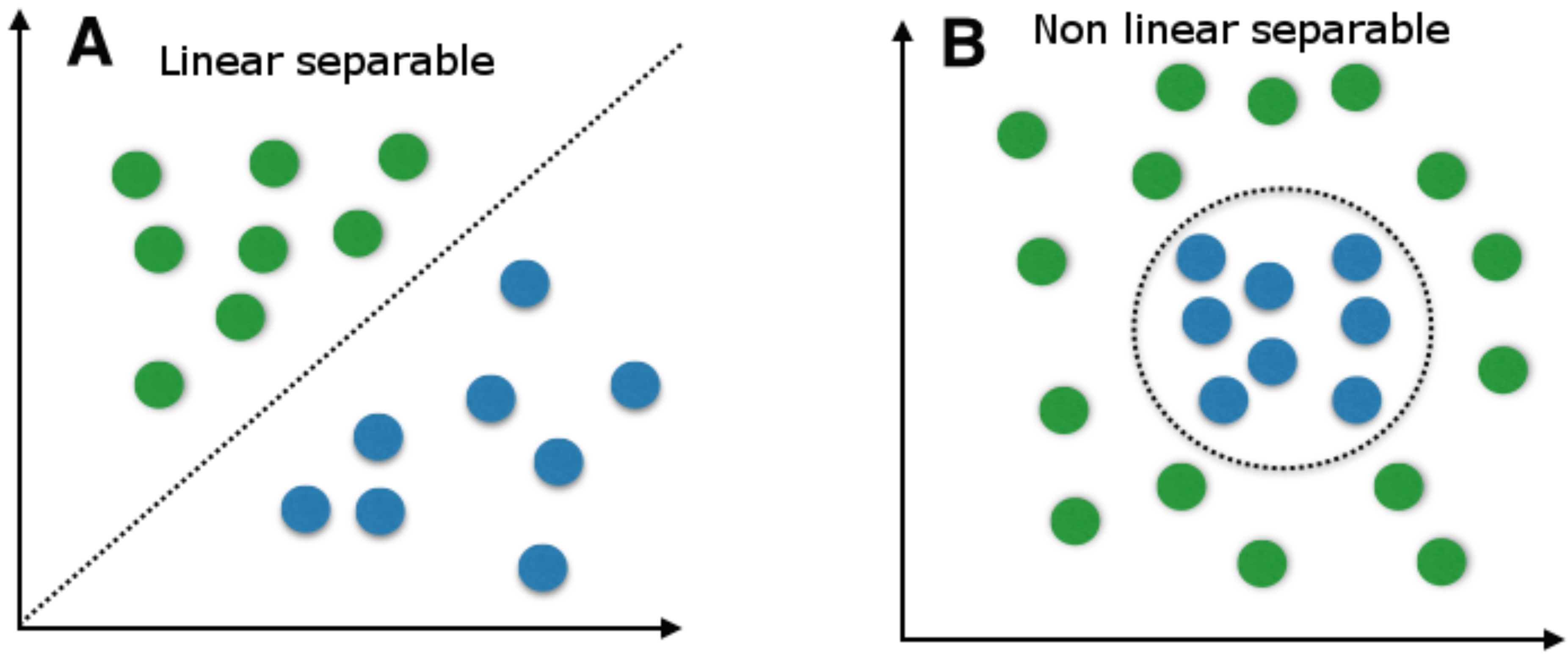
I love the states. I win them. Ohio is beautiful, I buy it. Thank you very much. I buy Hillary, it's beautiful and I'm happy about it.

DeepDrumpf @DeepDrumpf · 18h

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

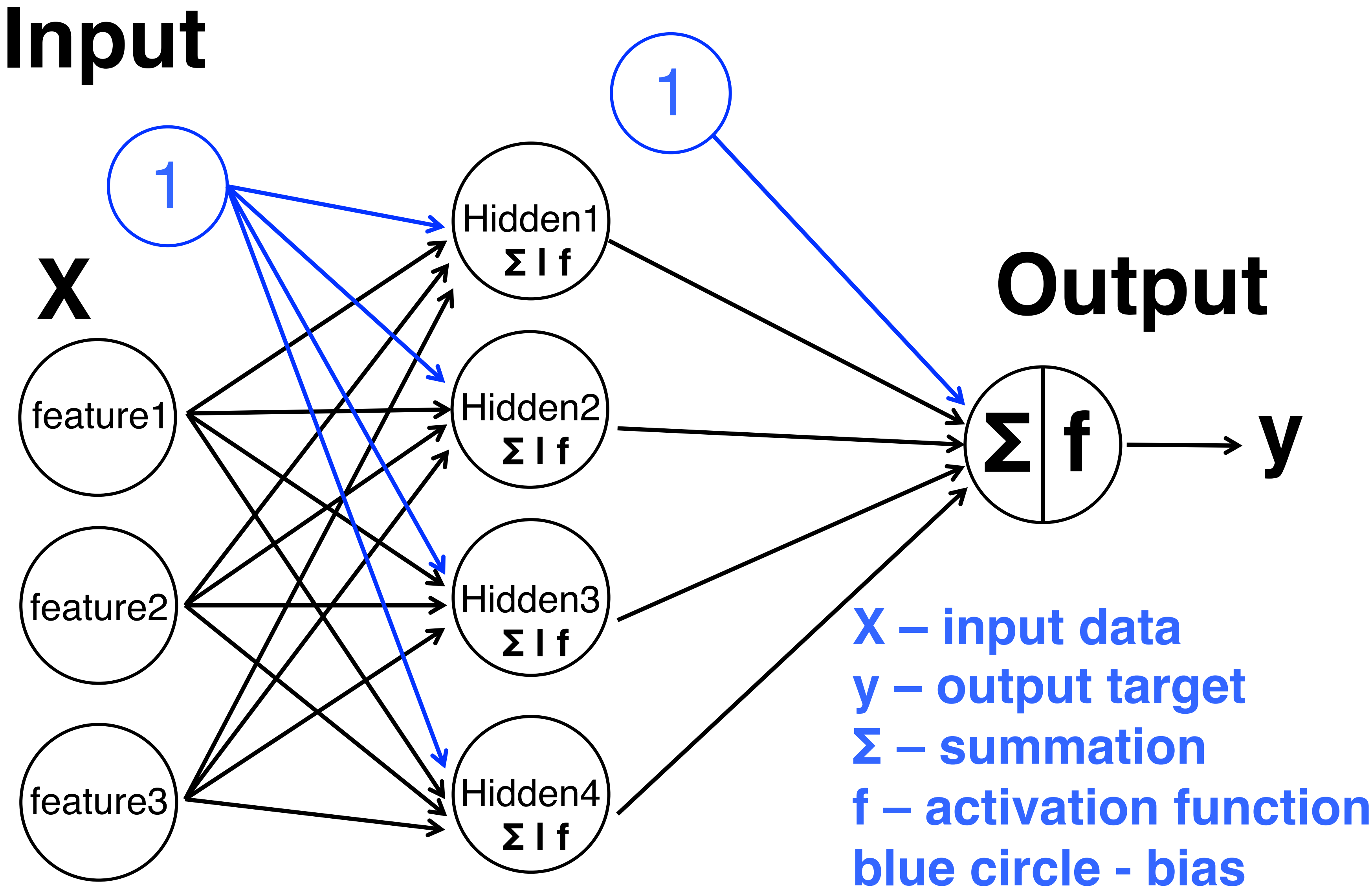
80 152

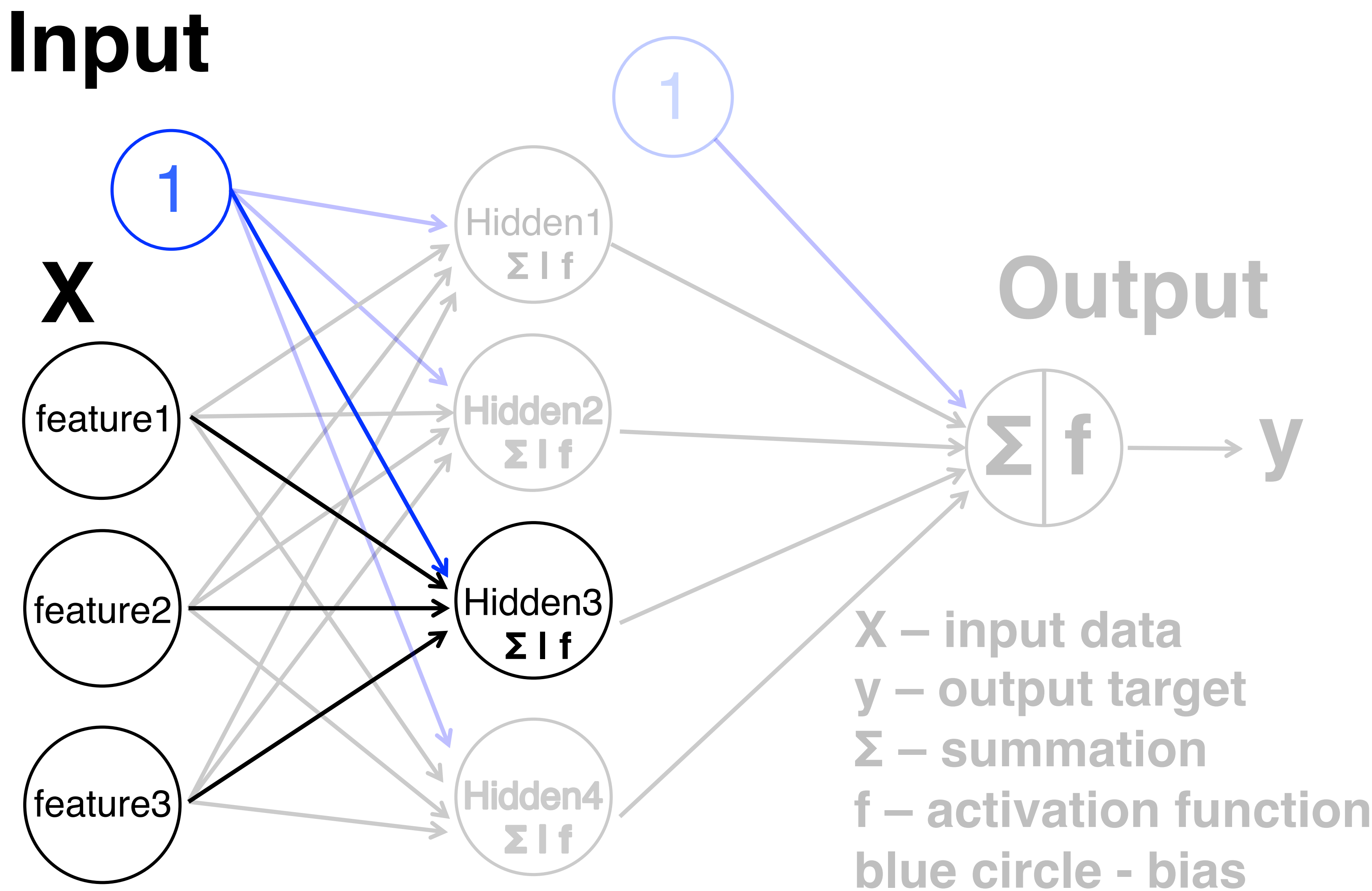
Perceptron limitations



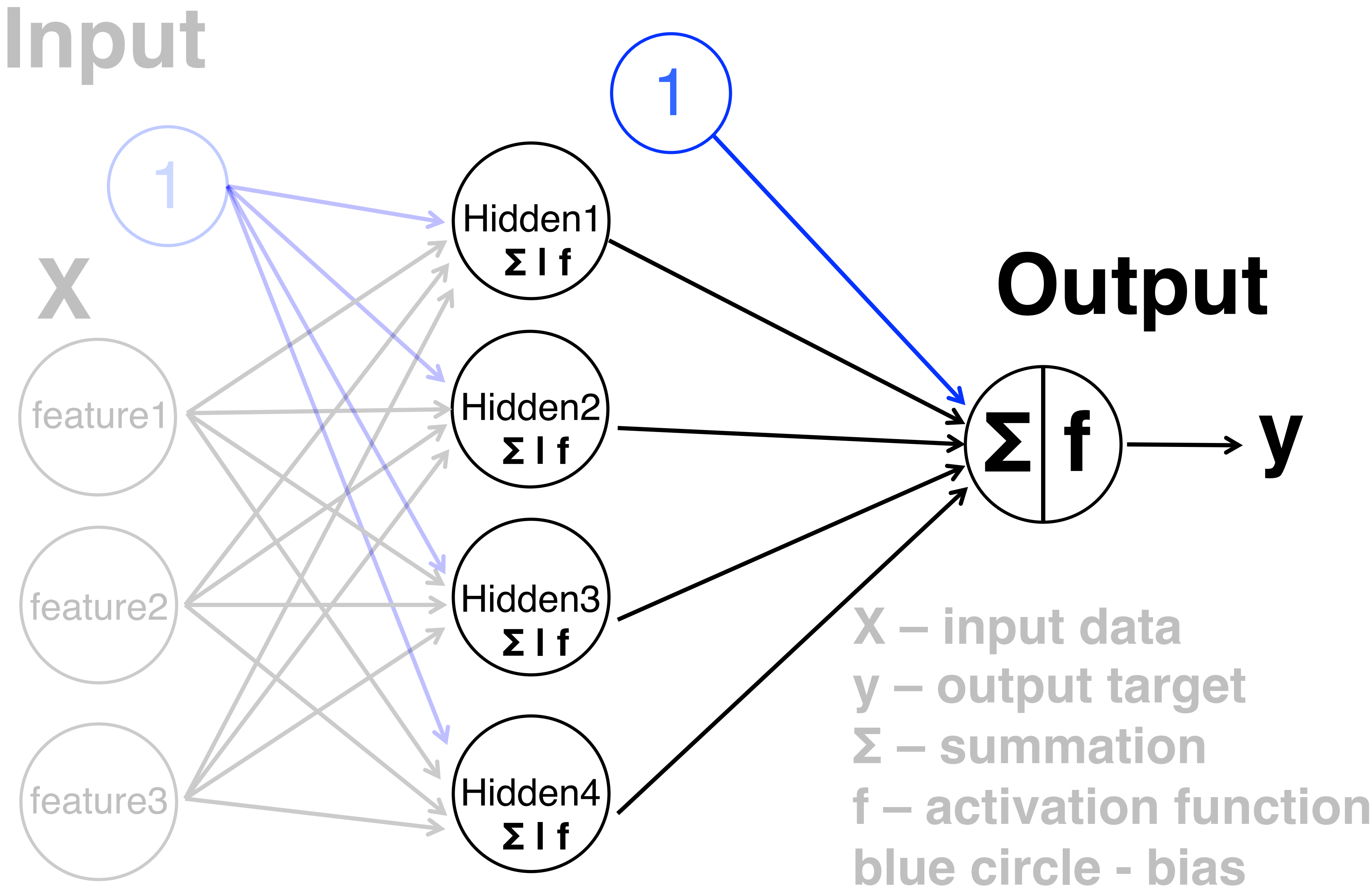

Winter of ANN

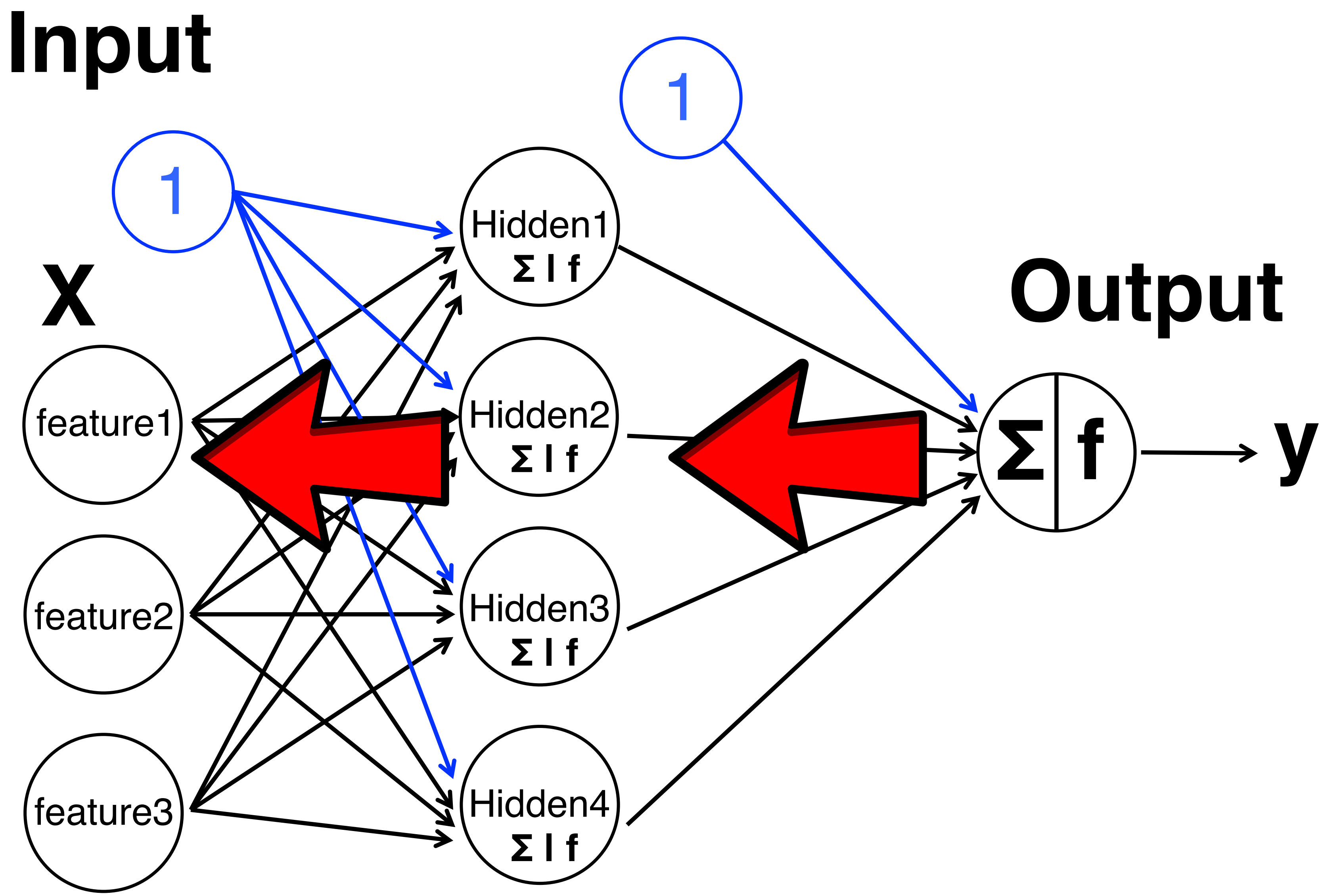
Multi-Layer Perceptron


Input
1
X
feature1
feature2
feature3
Hidden1
Σ | f
Hidden2
Σ | f
Hidden3
Σ | f
Hidden4
Σ | f
1
Output
Σ f
y
X – input data
y – output target
Σ – summation
f – activation function
blue circle - bias

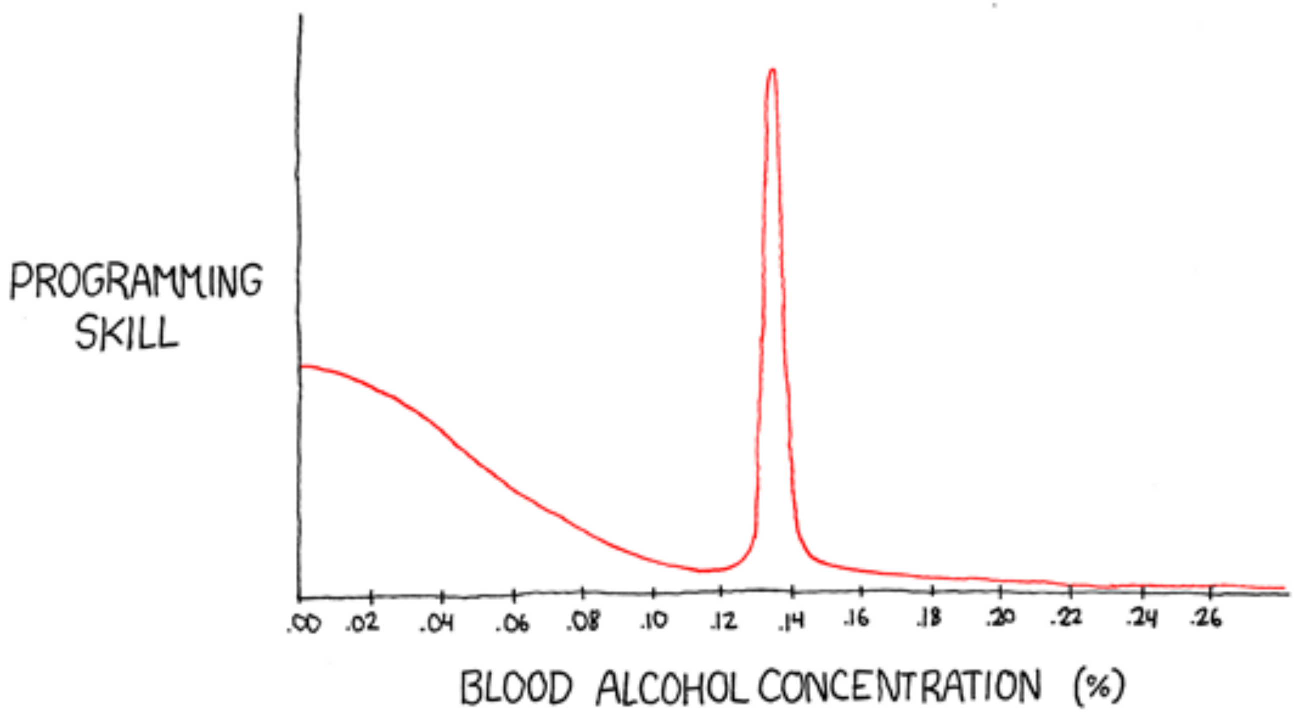


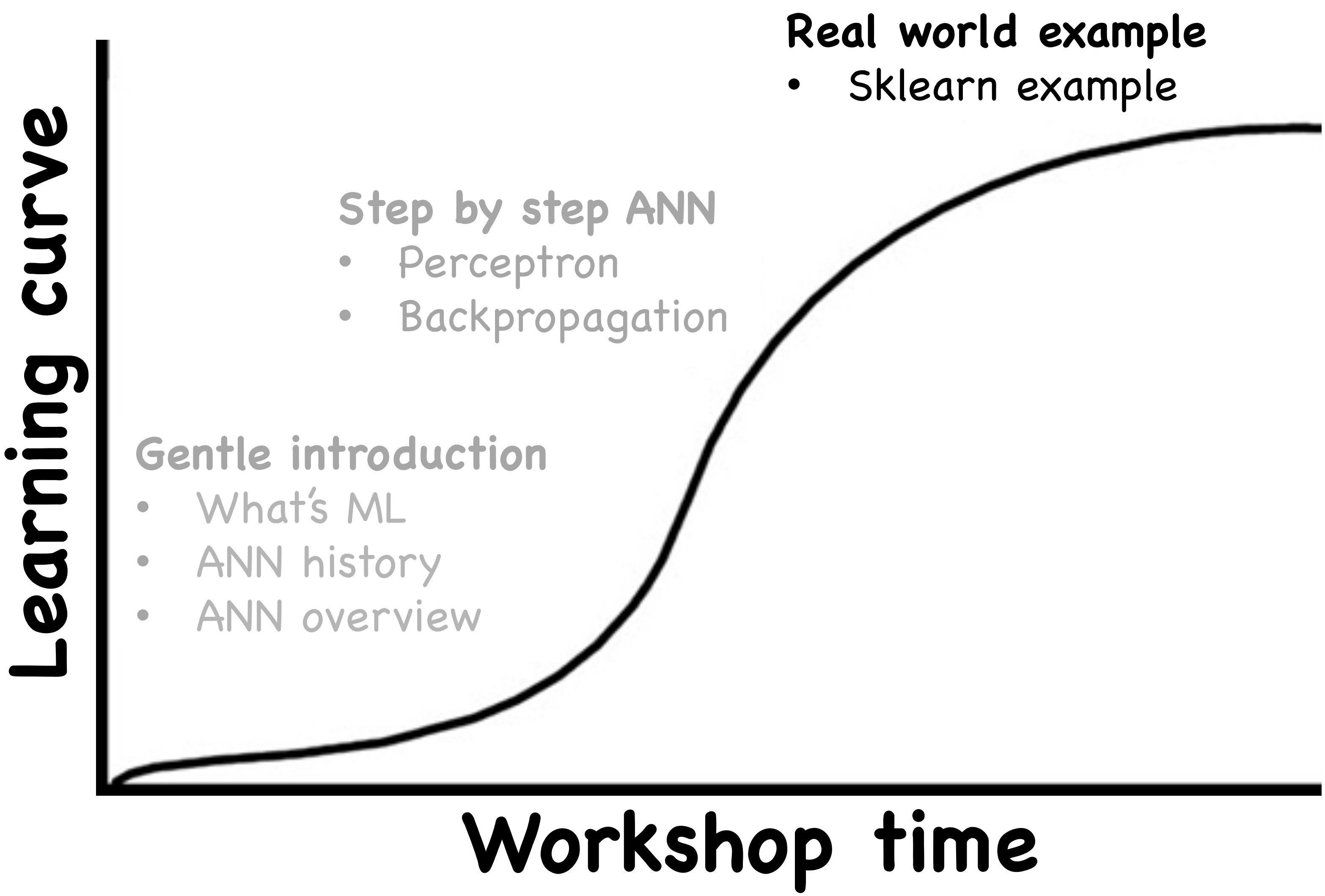
Input
1
X
feature1
feature2
feature3
Hidden1
Σ l f
Hidden2
Σ l f
Hidden3
Σ l f
Hidden4
Σ l f
1
Output
Σ f
y
X – input data
y – output target
Σ – summation
f – activation function
blue circle - bias


Input
1
X
feature1
feature2
feature3
Hidden1
Σ l f
Hidden2
Σ l f
Hidden3
Σ l f
Hidden4
Σ l f
1
Output
Σ f
y
X – input data
y – output target
Σ – summation
f – activation function
blue circle - bias

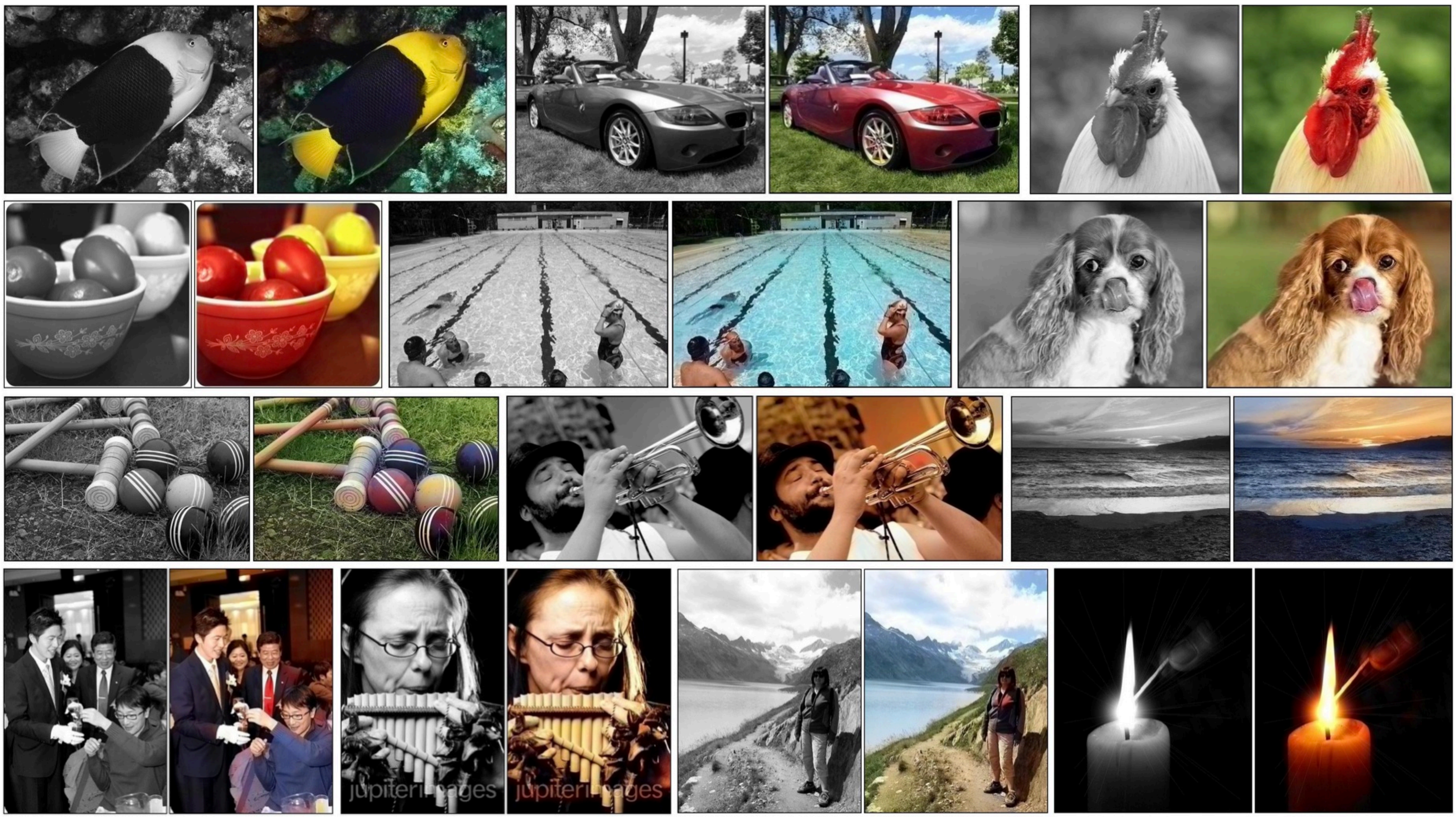

Input
X
1
feature1
feature2
feature3
Hidden1
Σ I f
Hidden2
Σ I f
Hidden3
Σ I f
Hidden4
Σ I f
1
Output
Σ
f
y

Go to notebook 02




Real world example
• Sklearn example
Step by step ANN
• Perceptron
• Backpropagation
Gentle introduction
• What's ML
• ANN history
• ANN overview
Learning curve
Workshop time

Application: Colourization



http://whattogive.com/videoColourization/

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

11

NEIL A. ARMSTRONG
JANET S. ARMSTRONG

2214

12/5 19 87

13-31
420

PAY TO THE ORDER OF Cuyahoga Valley Christian Academy $ 50 00

Fifty and xx/100 DOLLARS

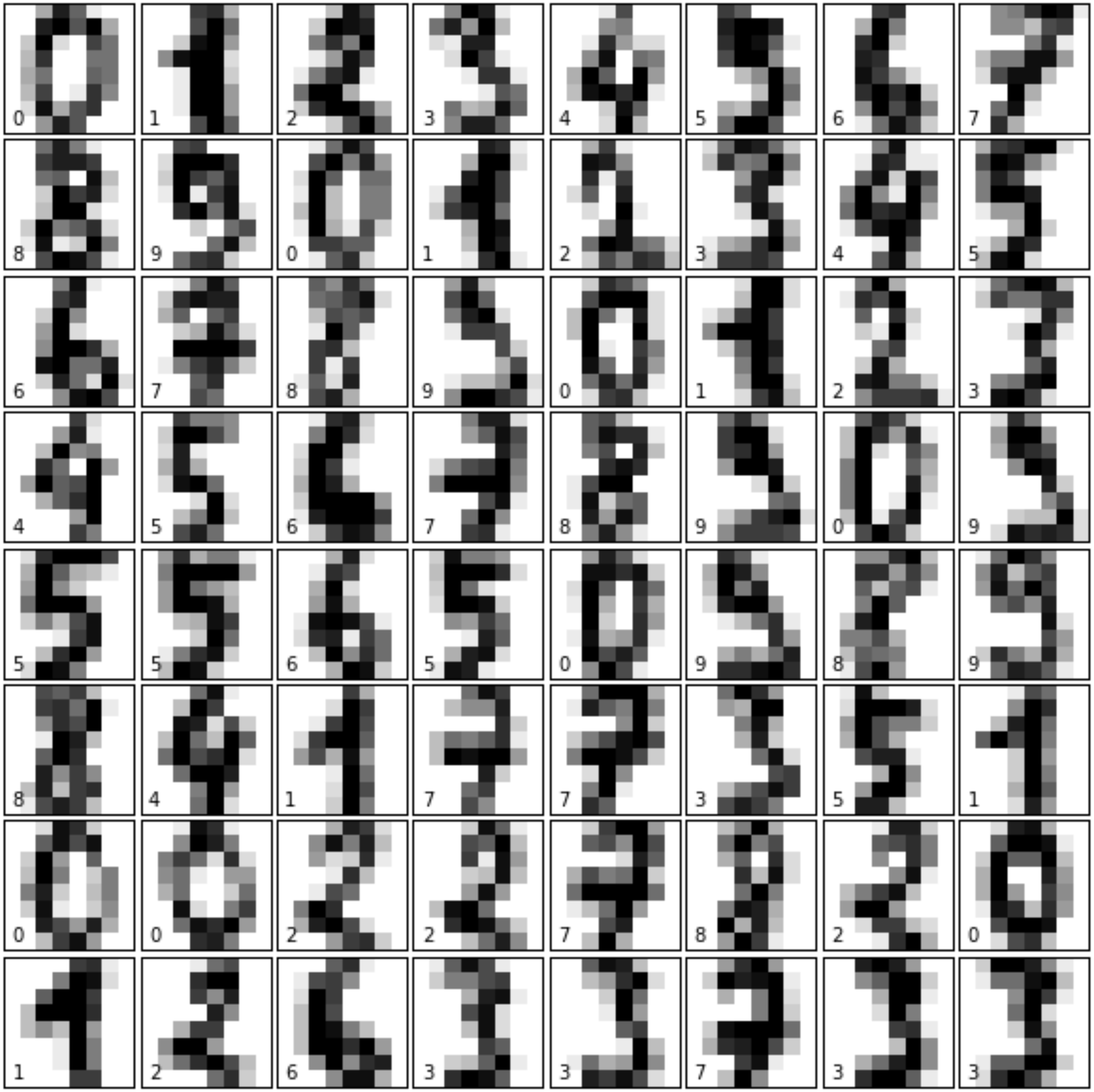
THE FIFTH THIRD BANK
CINCINNATI, OHIO 45201
FIFTH THIRD CENTER
38 FOUNTAIN SQUARE PLAZA, CINCINNATI, OH 45263

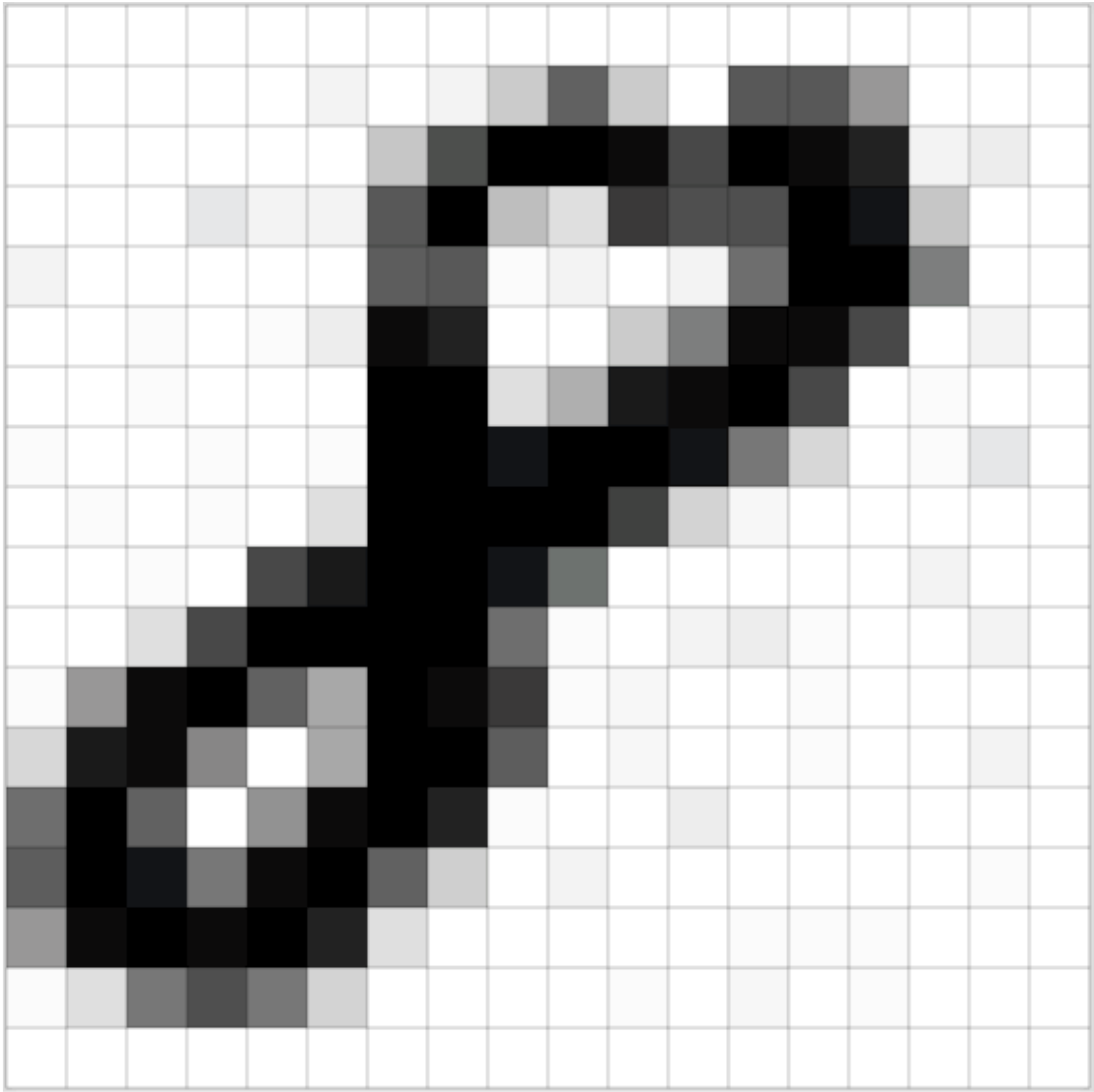
FOR ARMB

N. A. Armstrong

⑆042000314⑆ 2214 590⑈53609⑈ ⑈0000005000⑈

FALCONER • KY LITHO • KORB •

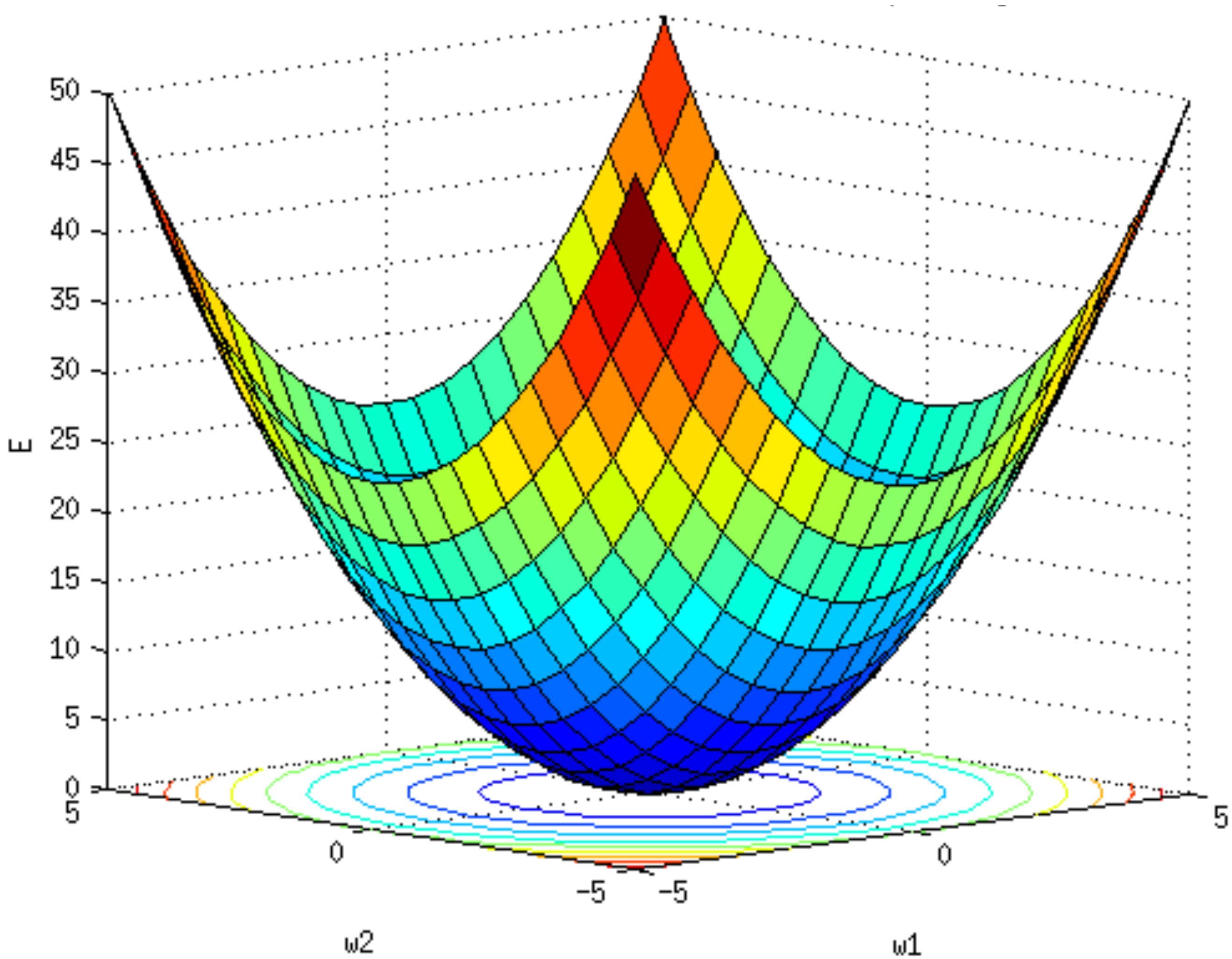

0 1 2 3 4 5 6 7
8 9 0 1 2 3 4 5
6 7 8 9 0 1 2 3
4 5 6 7 8 9 0 9
5 5 6 5 0 9 8 9
8 4 1 7 7 3 5 1
0 0 2 2 7 8 2 0
1 2 6 3 3 7 3 3



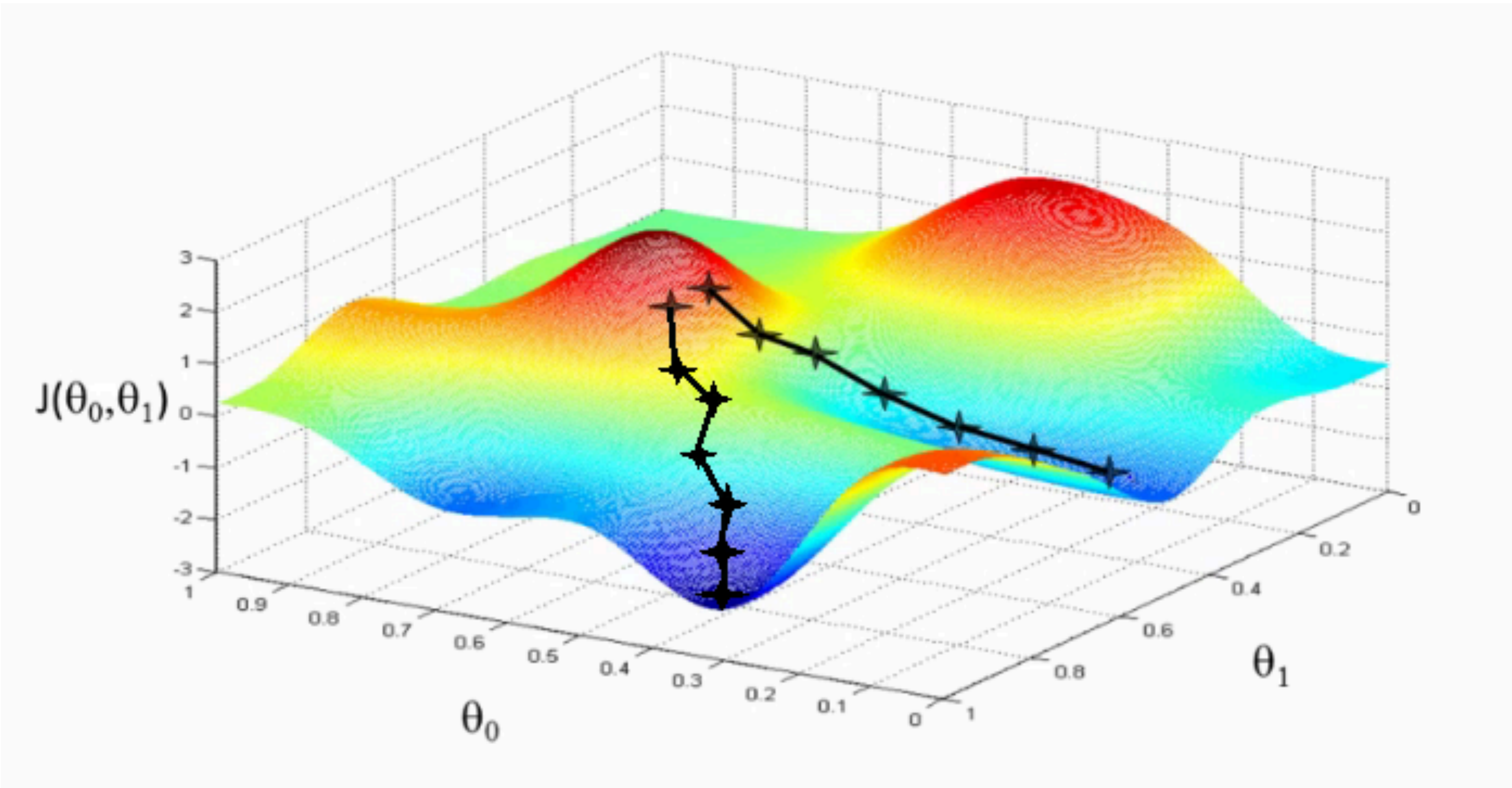
Go to notebook 03

More on ANN

Ideal Cost Function



Real-world Cost Function

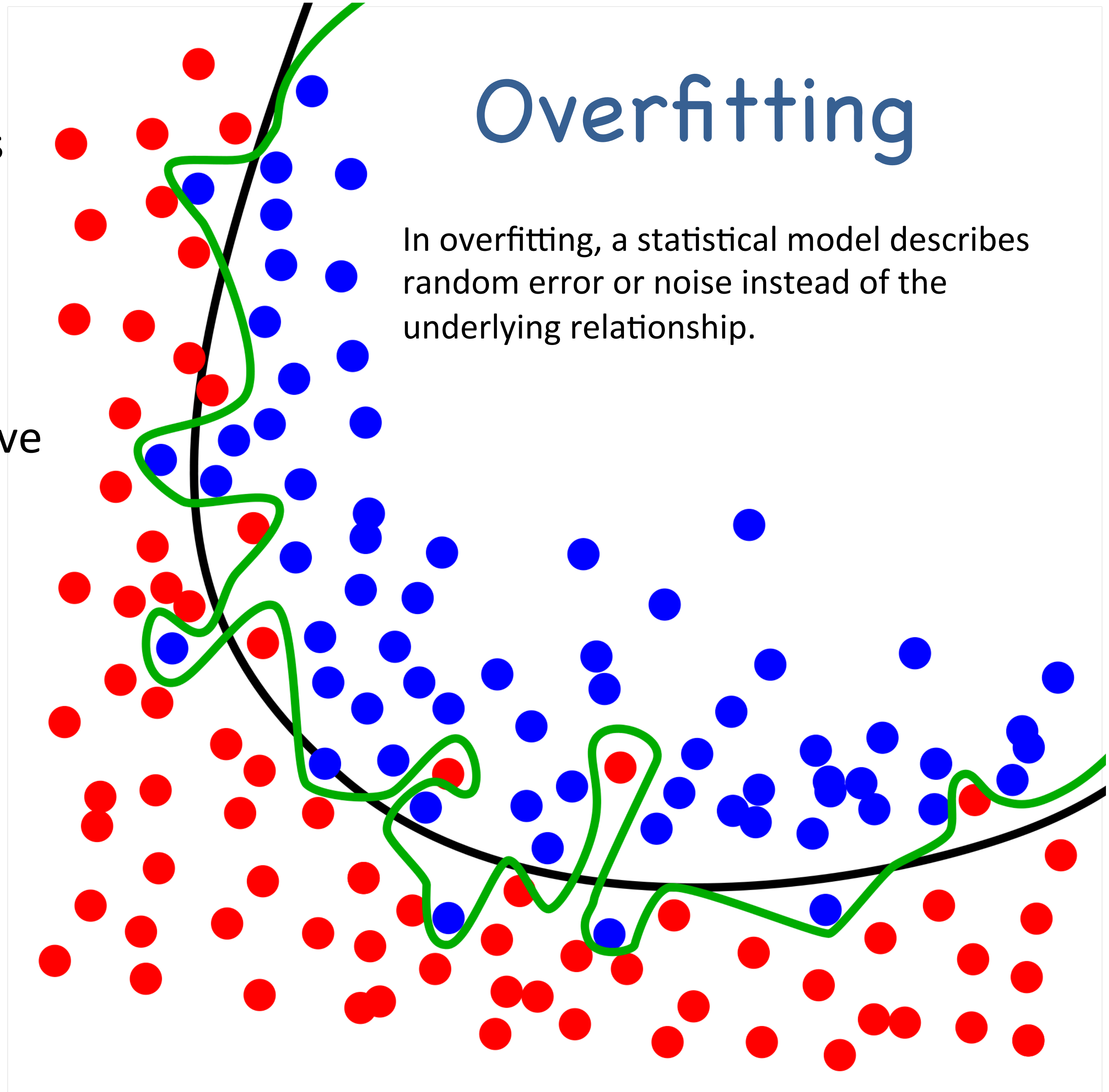


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

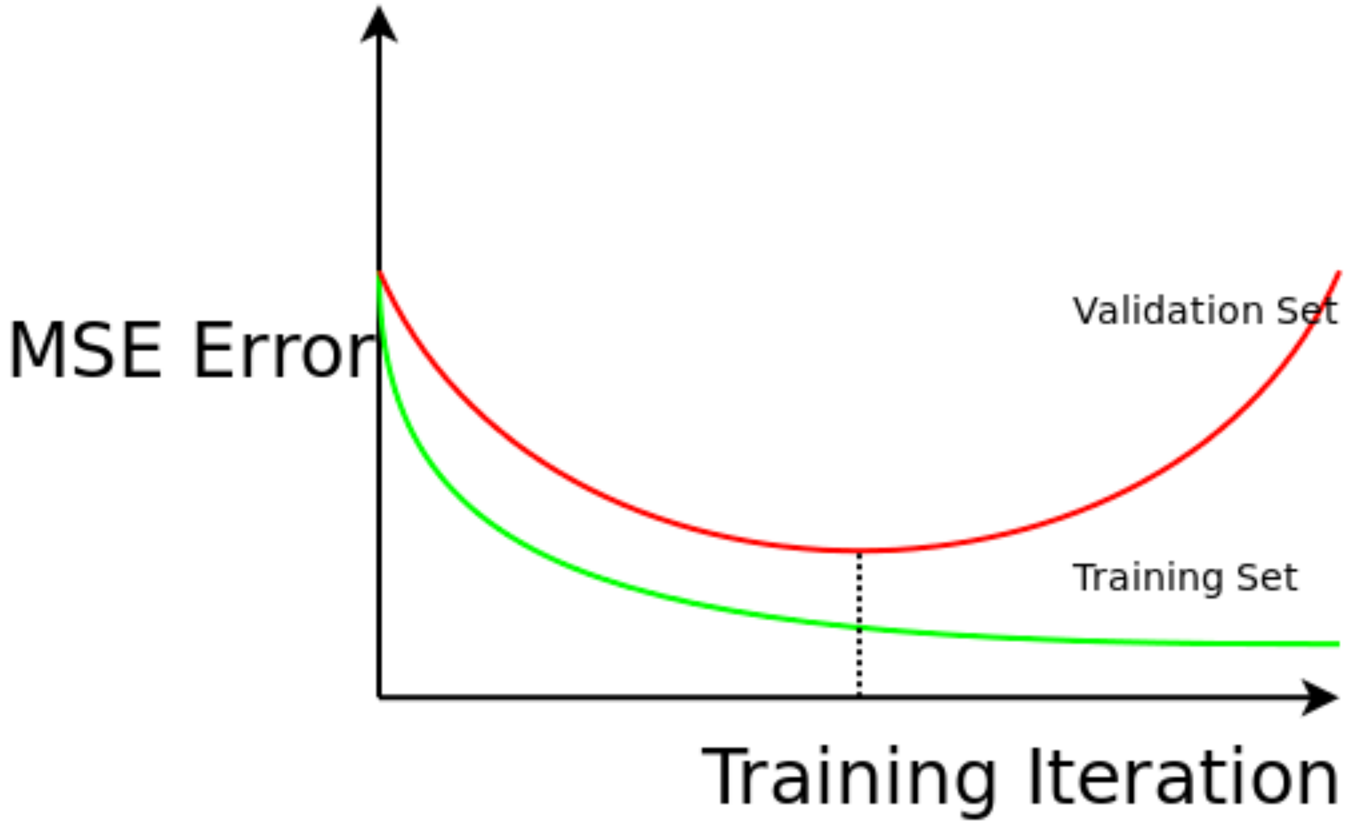
Overfitting

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.



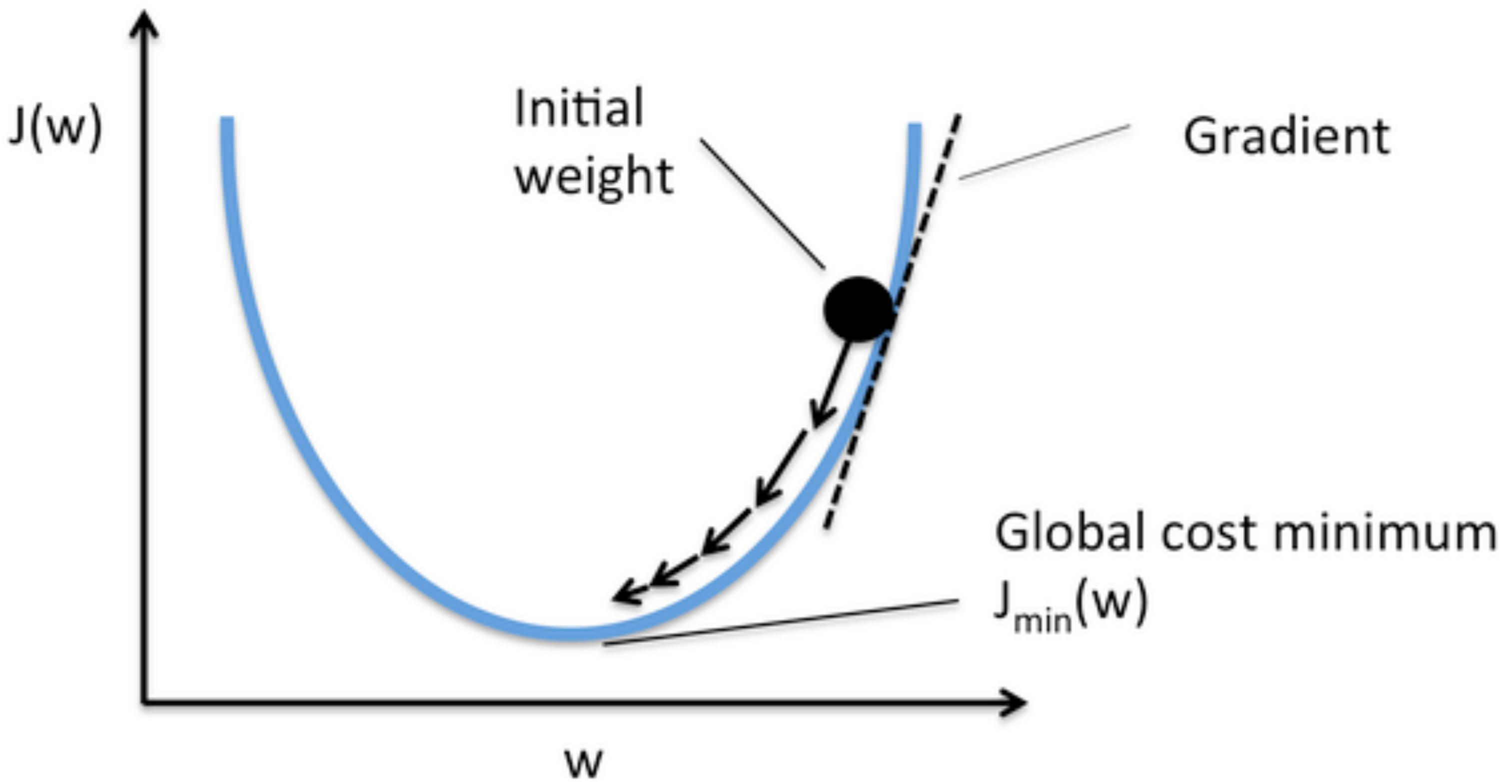
When to stop



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

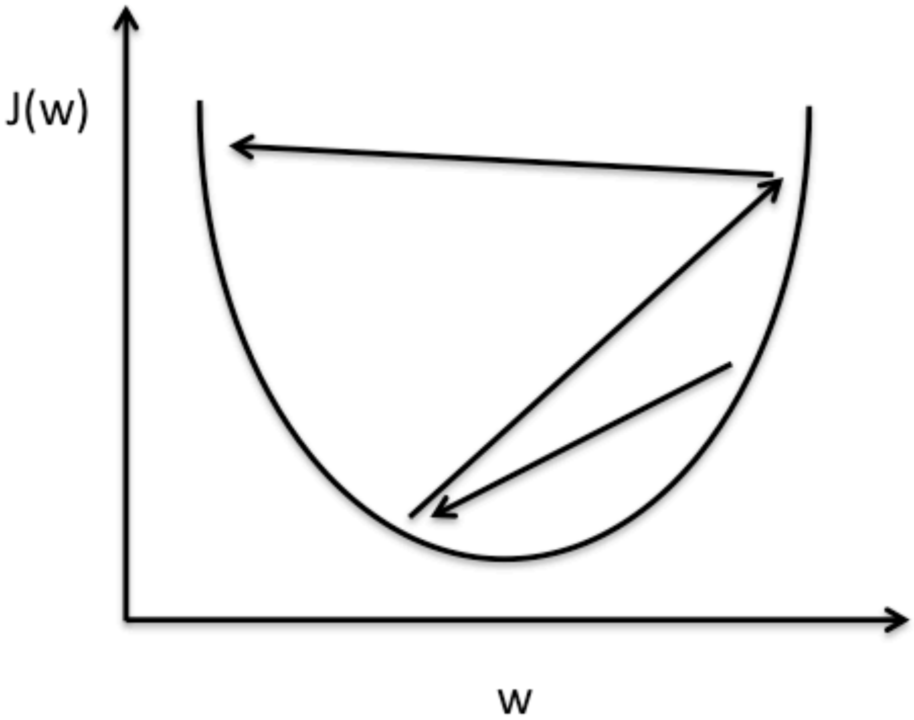
Validation error is the error when you get when you run the trained model on a set of data that it has previously never been exposed to.

Learning rate

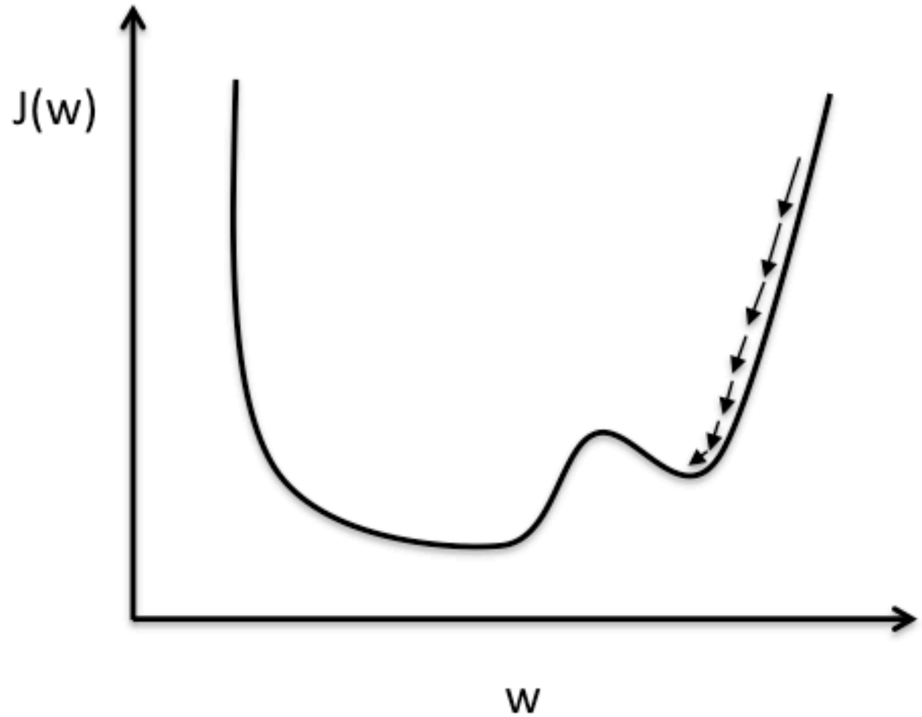


Learning rate controls how fast the model learns. The larger the learning rate, the faster it will learn, at the price of difficult to converge. Use an adaptive learning rate is better.

Learning rate

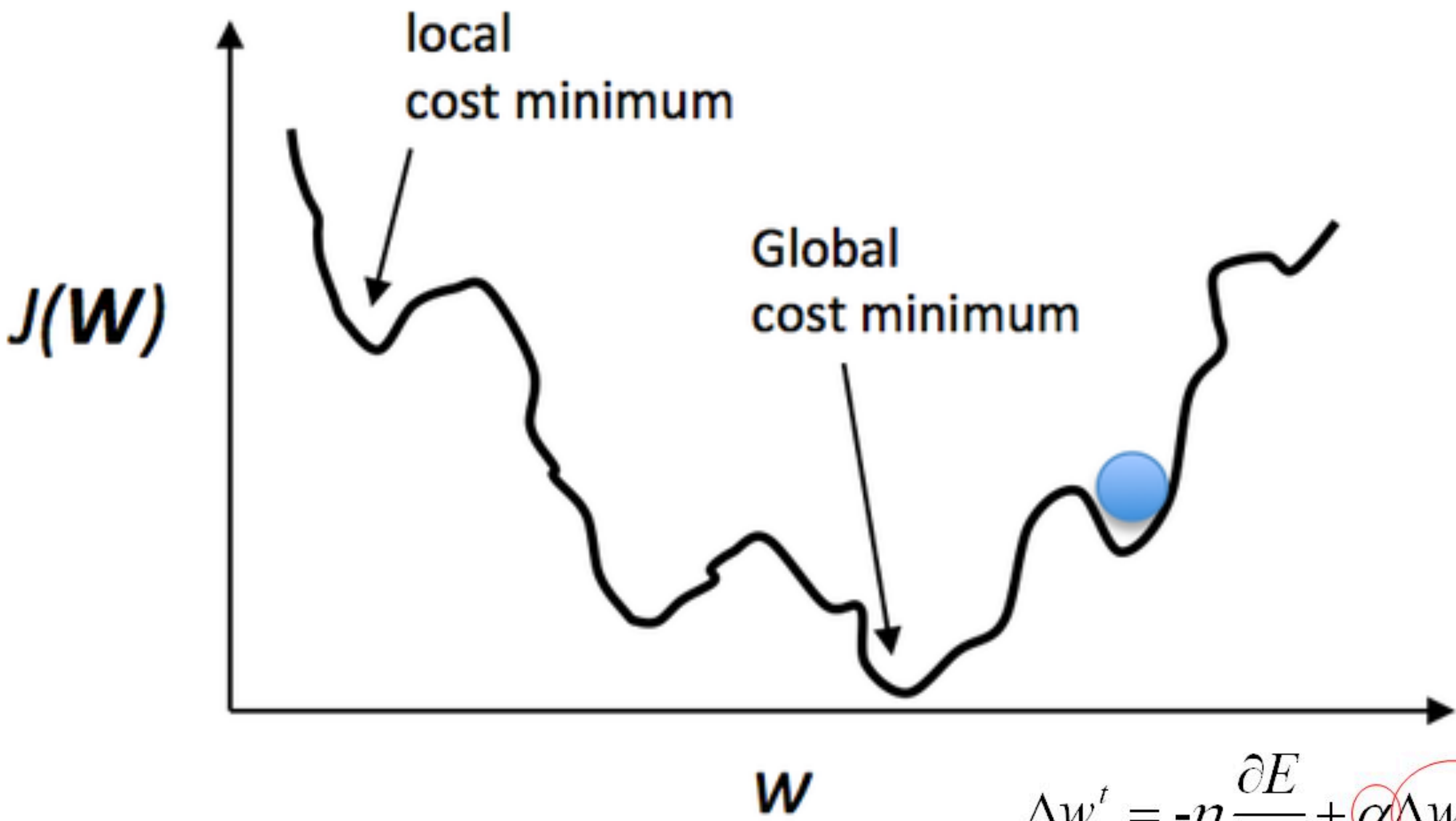


Large learning rate: Overshooting.



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

Momentum



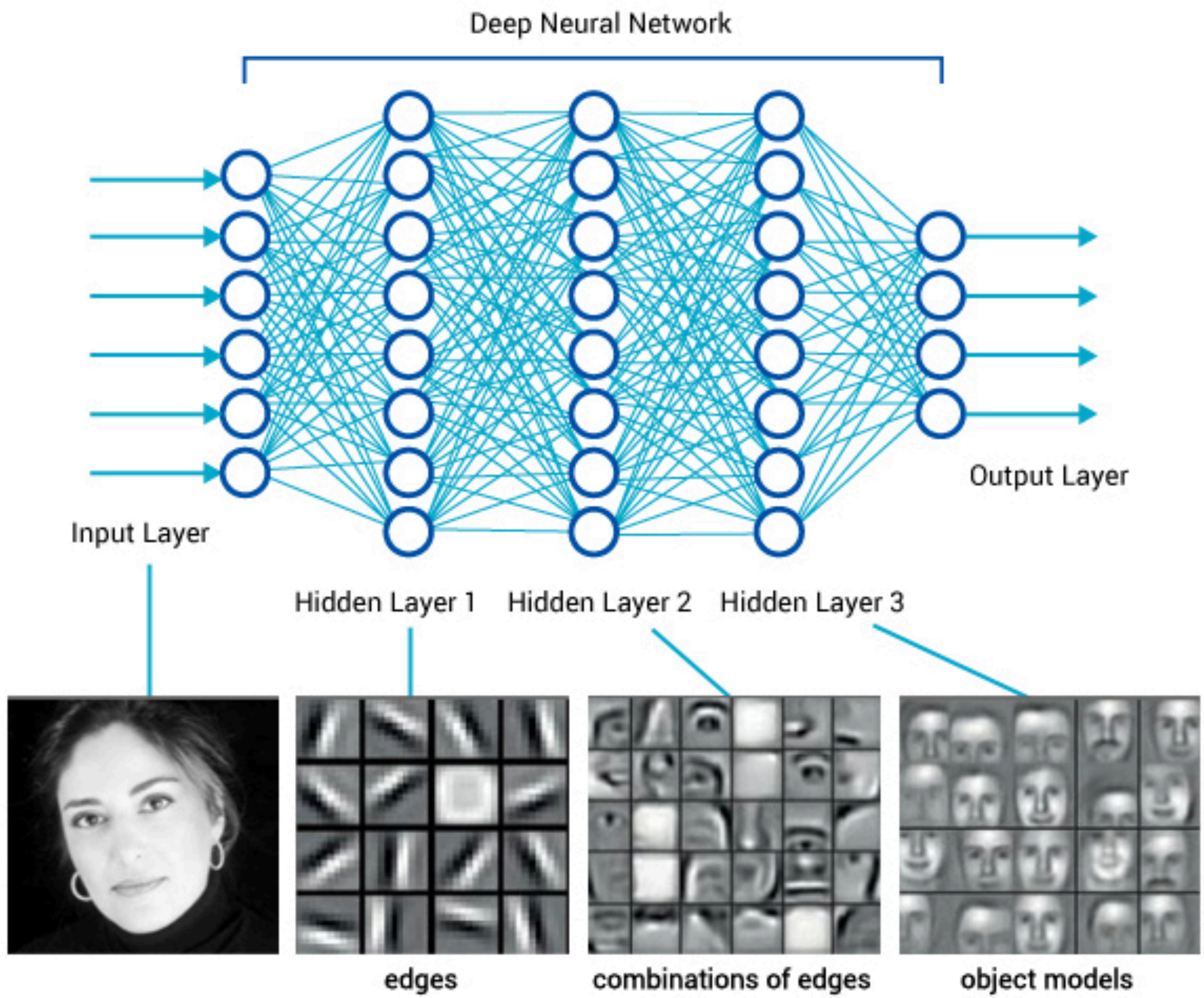
$$\Delta w^{t} = -\eta \frac{\partial E}{\partial w} + \alpha \Delta w^{t-1}$$

Momentum parameter
0.1-0.8

Momentum term

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


WE NEED TO GO
DEEPER


Deep Neural Network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
edges
combinations of edges
object models

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.


THANK YOU FOR LISTENING
ANY QUESTIONS?