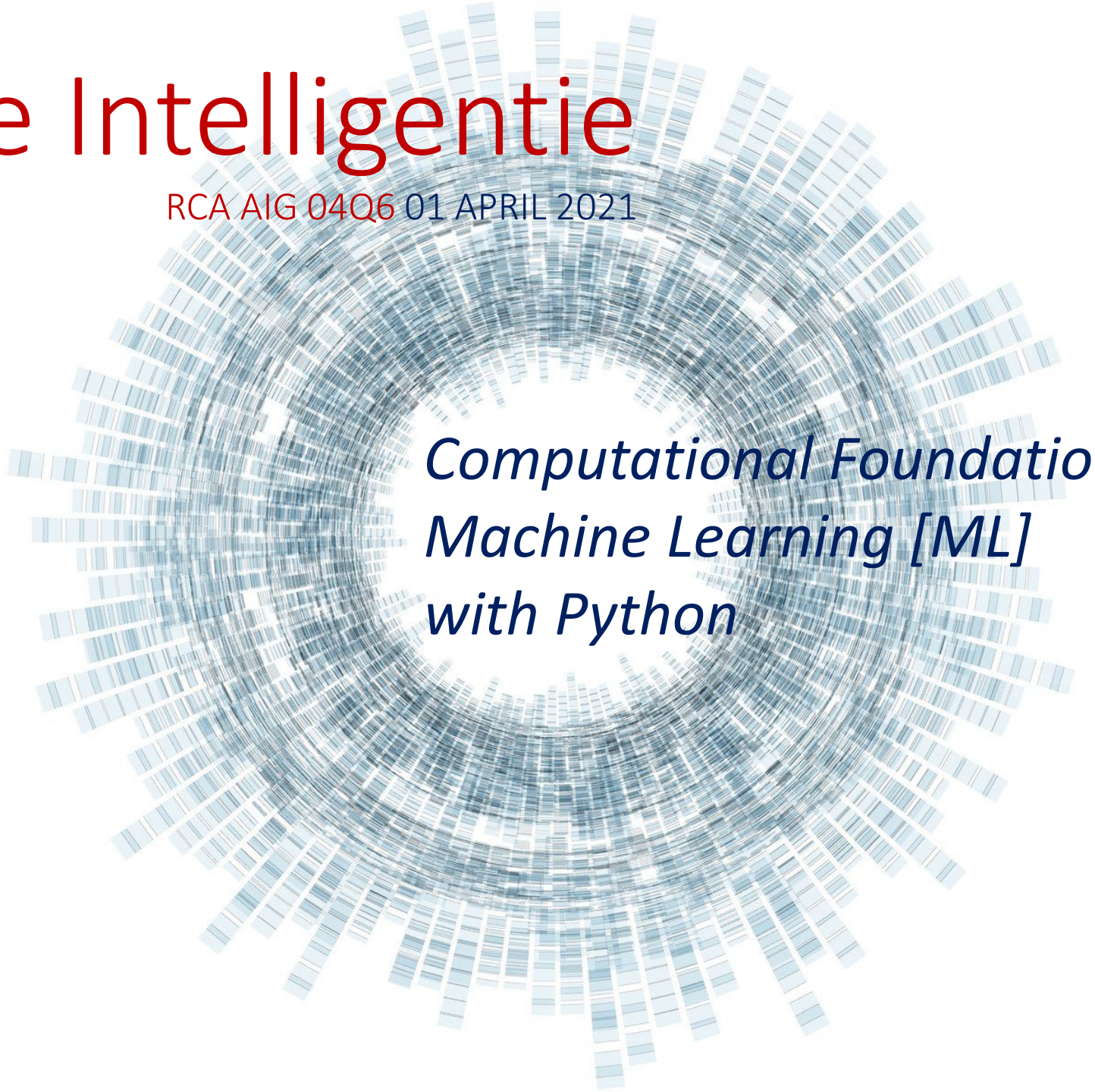
Artificiële Intelligentie

RCA AIG 04Q6 01 APRIL 2021

HOGESCHOOL ROTTERDAM



Computational Foundations of Machine Learning [ML] with Python

DataLab

Rob van der Willigen

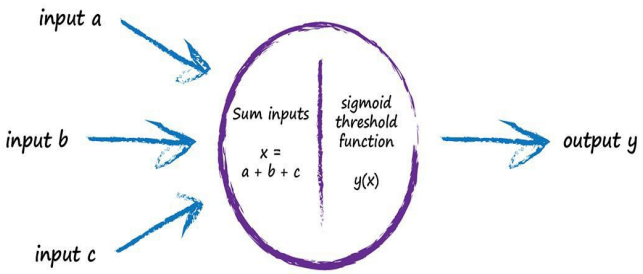
CONTEXT

- Prerequisites: basics in linear algebra, probability, and analysis of algorithms.
- Workload: homework assignments
- GitHub: Start a ML repository at GitHub

Lecture 03

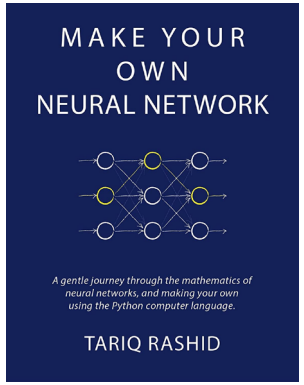
- Basic definitions and concepts of Machine Learning (ML) PART02.
- How to get from ML concepts & Models to Python code.

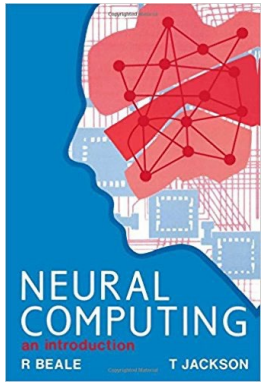
THE DOT PRODUCT

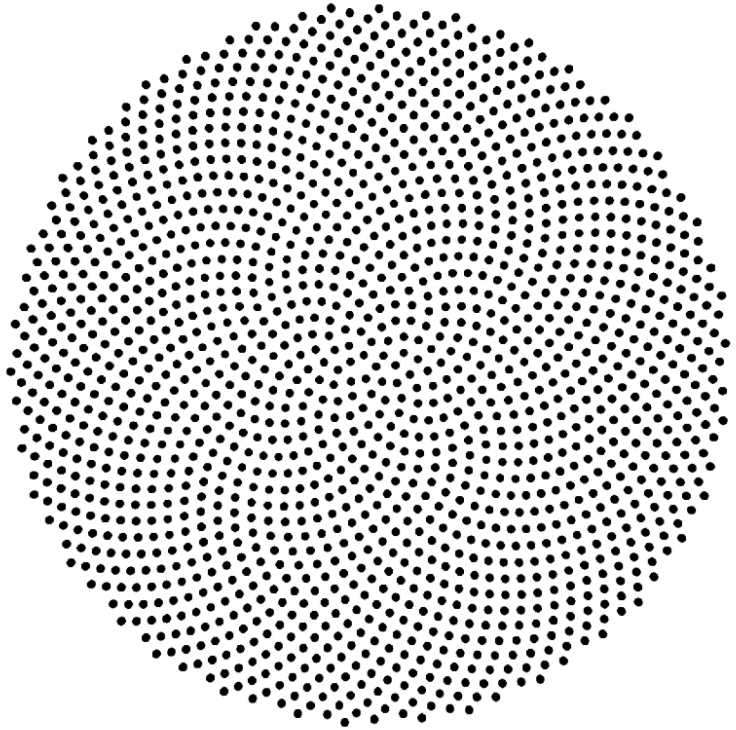



```python
import numpy as np
import matplotlib.pyplot as plt

class Perceptron(object):

    def __init__(self, no_of_inputs, no_of_training_epochs=20, learning_rate=0.01):
        self.no_of_training_epochs = no_of_training_epochs
        self.learning_rate = learning_rate
        self.weights = np.random.normal(0, 10, no_of_inputs + 1)
        self.bias = self.weights[0]
        self.epoch_list=[]
        self.error_history=[]


    def step_function(self, inputs):
        netto_summation = np.dot(inputs, self.weights[1:]) + self.bias
        if netto_summation > 0:
          activation = 1
        else:
          activation = 0
        return activation

    def train(self, training_inputs, teacher_labels):
        epoch =0;
        i =0;
        for _ in range(self.no_of_training_epochs):
          epoch+=1
          for inputs, teacher in zip(training_inputs, teacher_labels):
              activated_perceptron_output = self.step_function(inputs)
              error = (teacher - activated_perceptron_output)
              self.weights[1:] += self.learning_rate * (error) * inputs
              self.bias += self.learning_rate  * error
              #print(teacher, activated_perceptron_output)
              i+=1
              self.epoch_list.append(epoch)
              self.error_history.append(np.sum(np.abs(error)))
              #print("iteration = ", i, " epoch = ", epoch, ' error: ', error, '  weights: ', self.weights[1:])
```

{01}
Fundamentals







HOGESCHOOL
ROTTERDAM

Key-words / Concepts / Labels

Artificial neuron === Model of single biological neuron / Perceptron

Feedforward / Feed-backward / Hebbian learning

Bias / Threshold

Weights /adjustable parameters / Memory

Unit / Node / Activation Functions (Sigmoid / Step)

Input / Output / Layers / Input vs Output Space

Target / Teacher / Error / learningrate

Iterations / Epochs / Baches

Logical functions AND / OR / XOR (Boolean Algebra)

Maths + Python Key-words / Concepts

HOGESCHOOL ROTTERDAM

PYTHON MODULES

```
import Numpy as np
import Matlibplot.pyplot as plt
```

class
def return
for (loops)
if else
zip
print
list
append

+=

Matrix Calculus

np.zeros
np.random.normal
np.dot

Plotting

```
plt.subplots()
plt.subplot(221)
plt.plot(epoch, error)
plt.xlabel('Epoch')
plt.ylabel('Error')
```

The PERCEPTRON

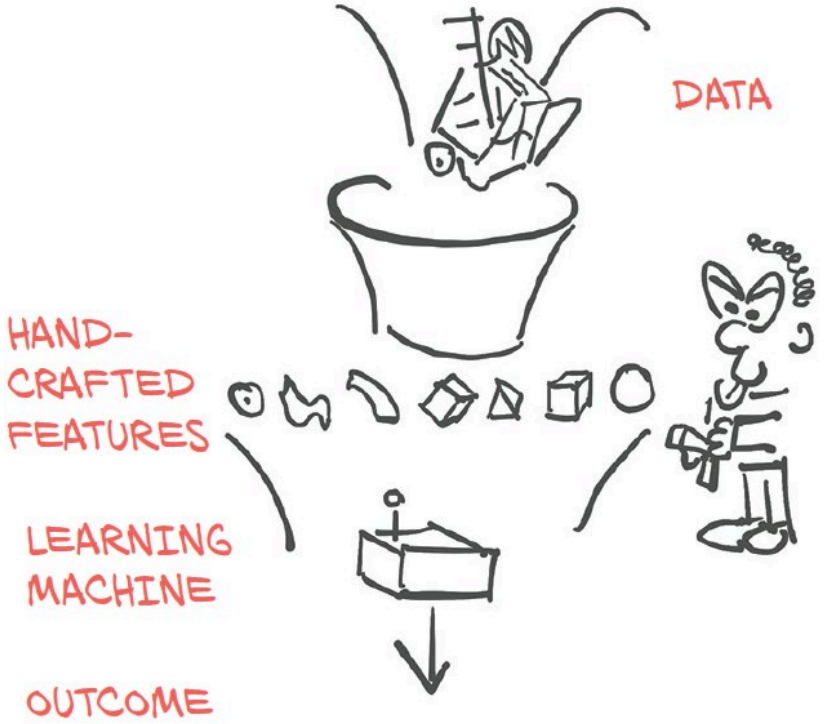
The perceptron was very promising, but it was soon discovered that is has serious limitations as it only works for linearly-separable classes.

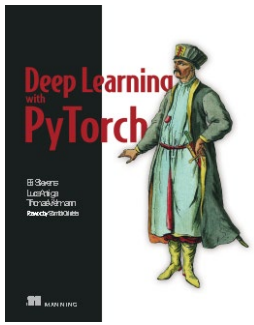
In 1969, Marvin Minsky and Seymour Papert demonstrated that it could not learn even a simple logical function such as XOR.

This led to a significant decline in the interest in perceptron's and Machine learning as a whole.

MACHINE LEARNING [ML]

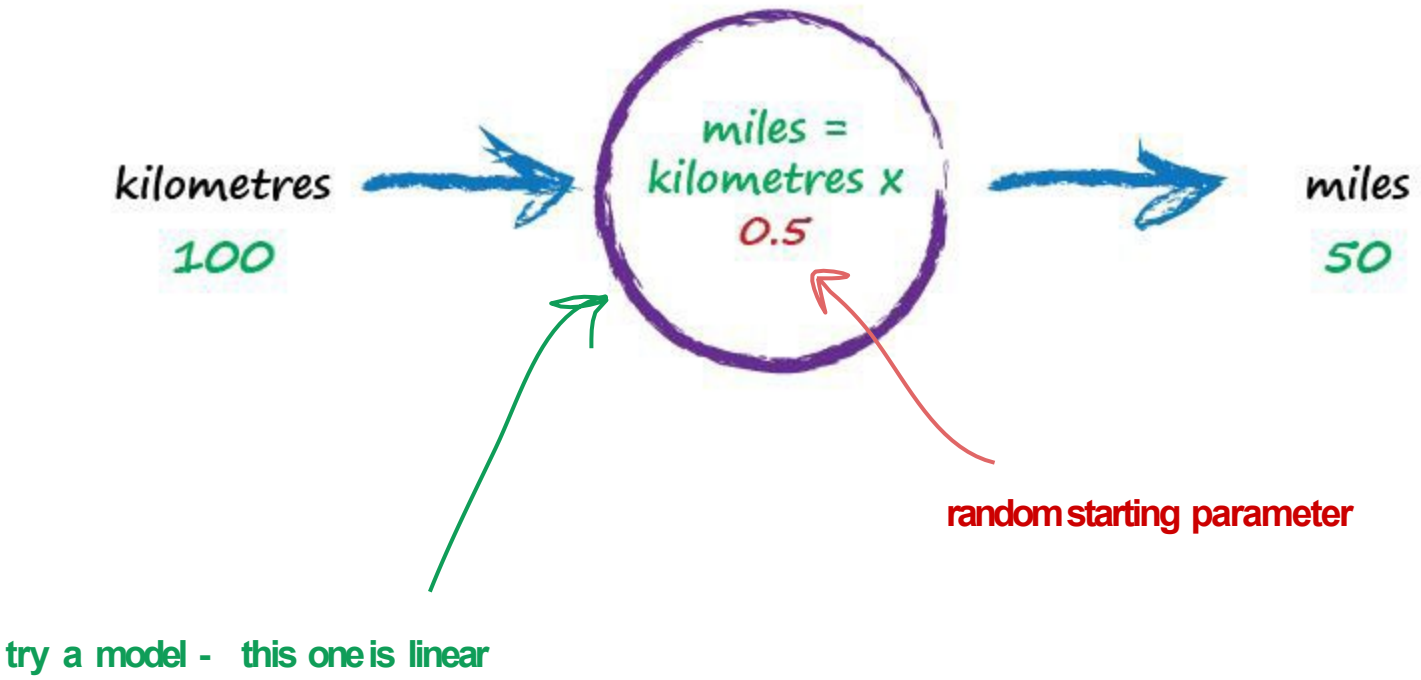
HOGESCHOOL ROTTERDAM



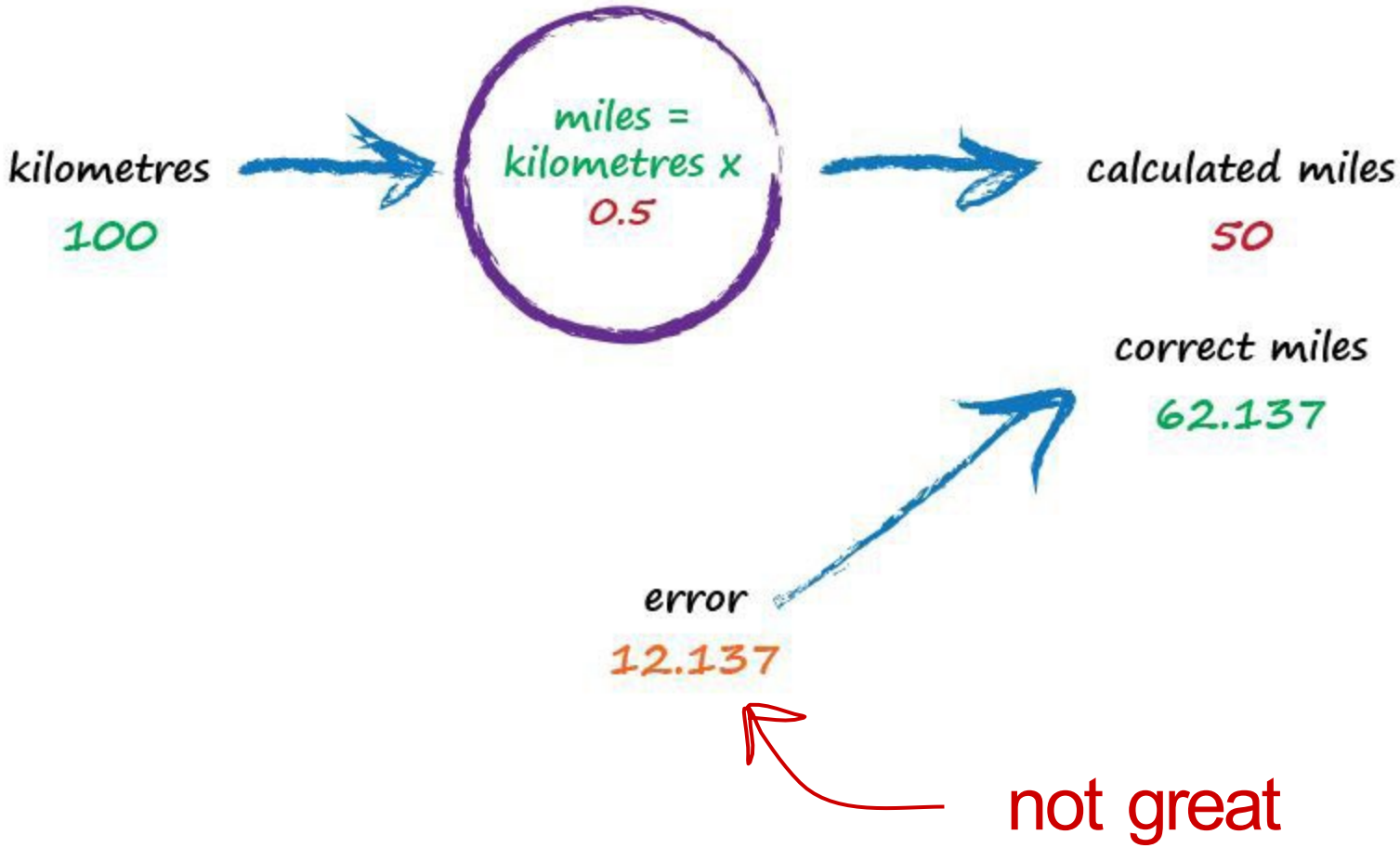


The Concept of Machine Learning

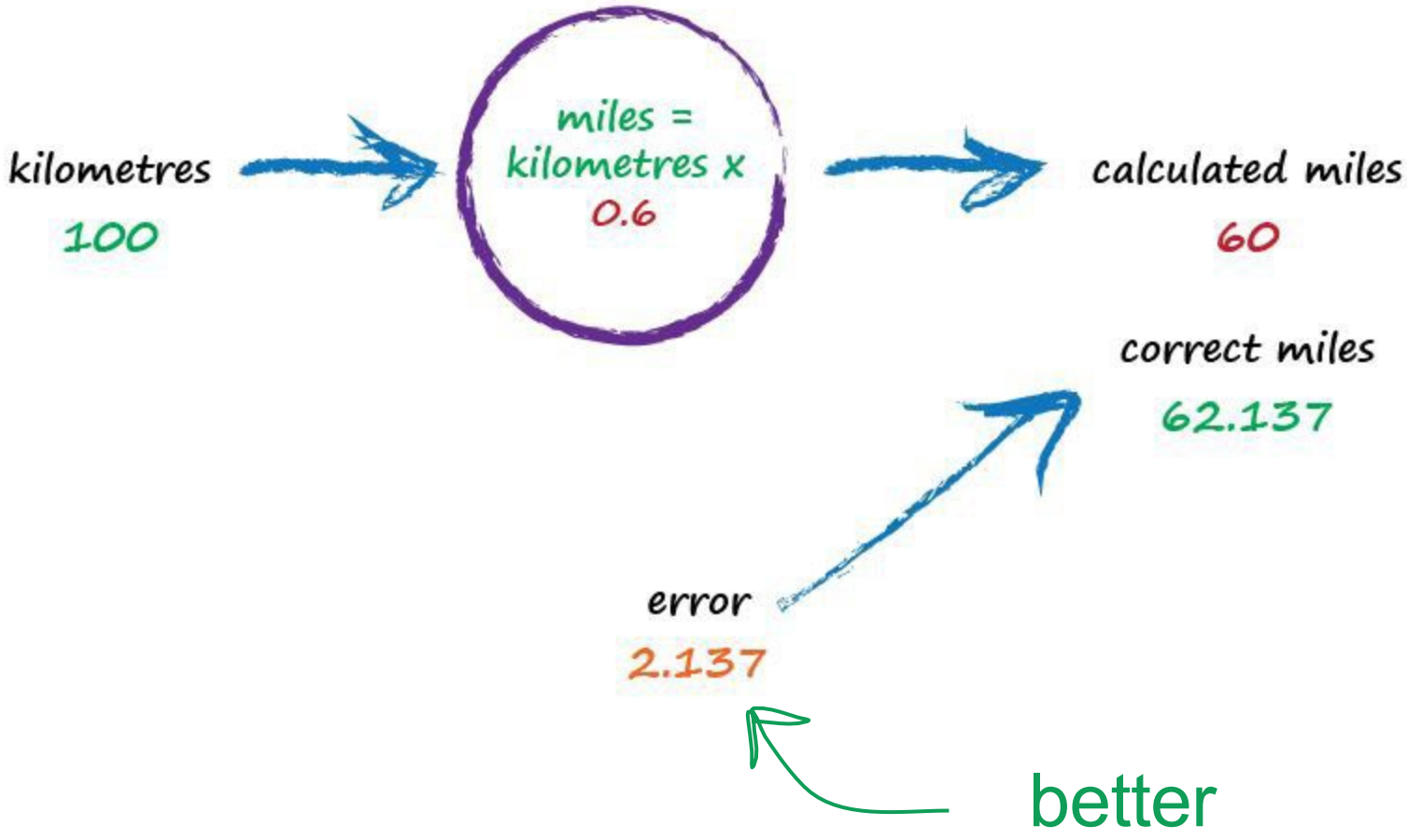
The Concept of Machine Learning



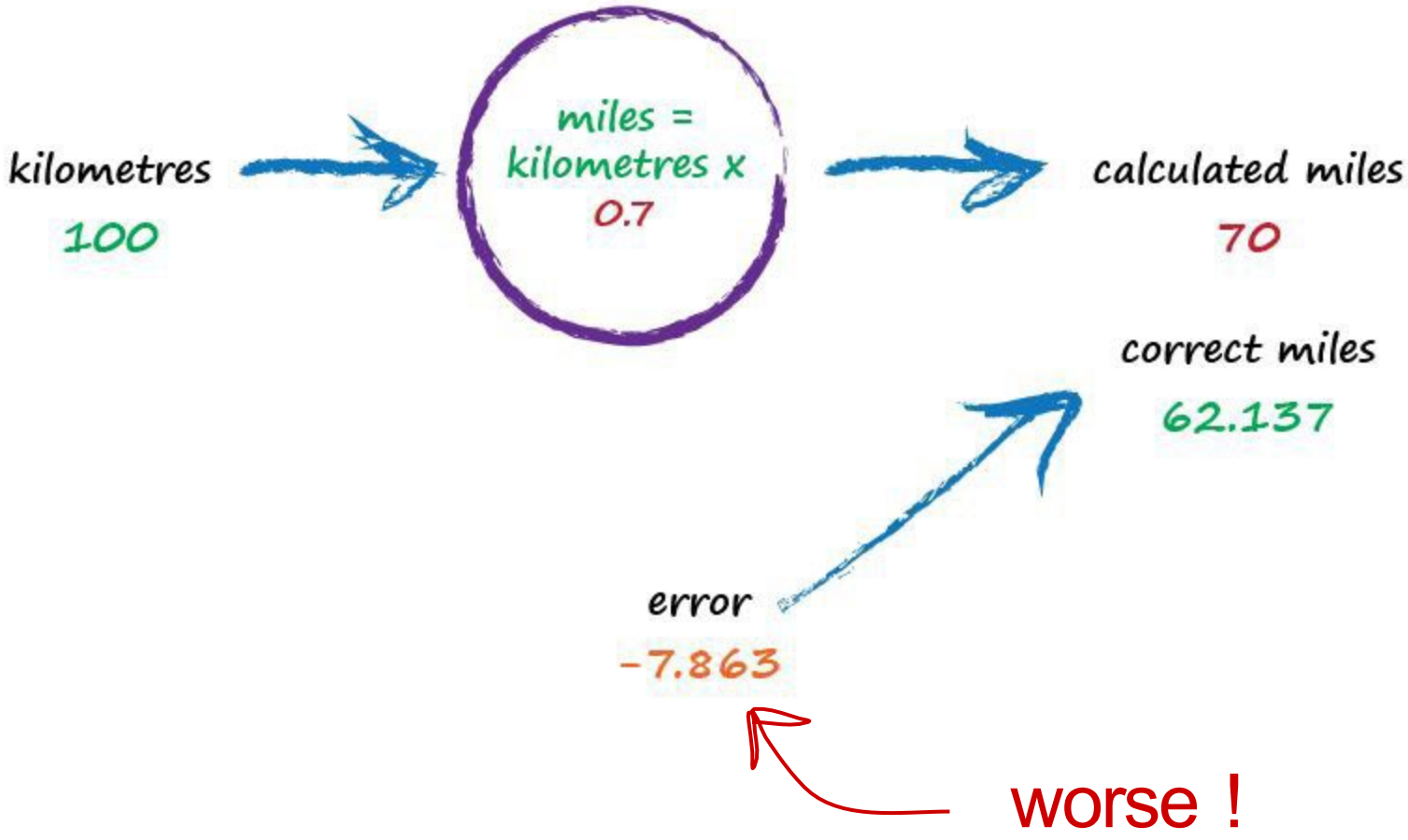
The Concept of Machine Learning



The Concept of Machine Learning



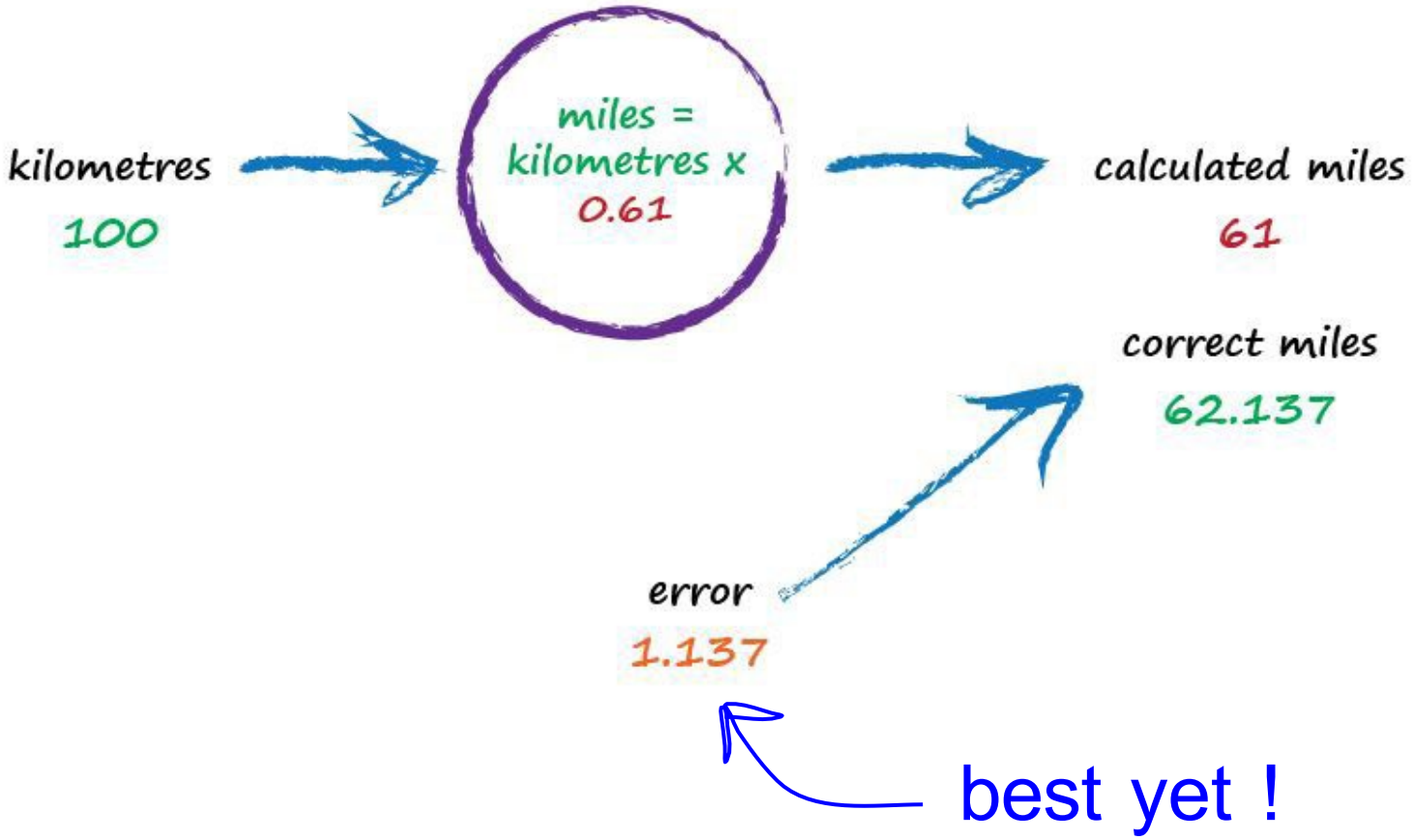
The Concept of Machine Learning



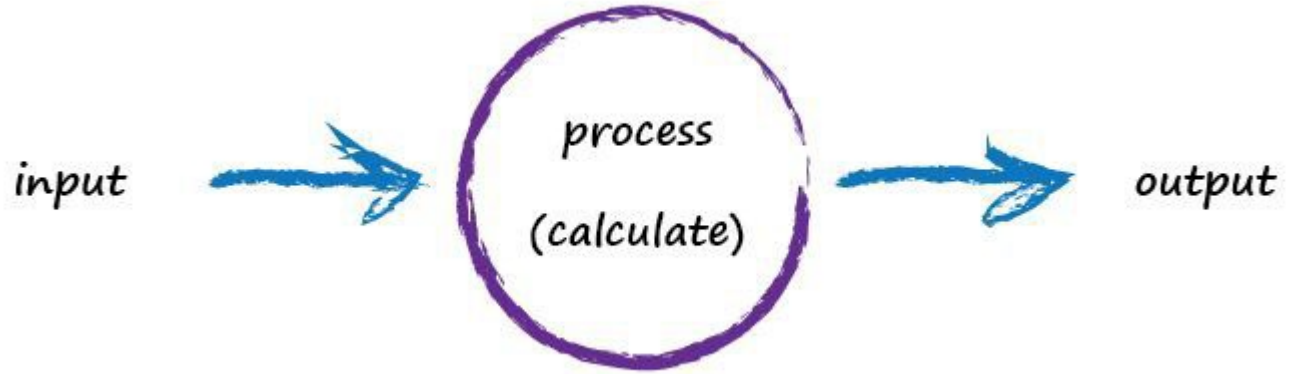
Basic Concept of Machine Learning (ML)

is iterative in nature;

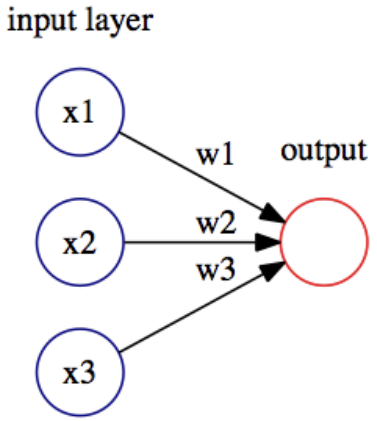
requires a teacher & memory

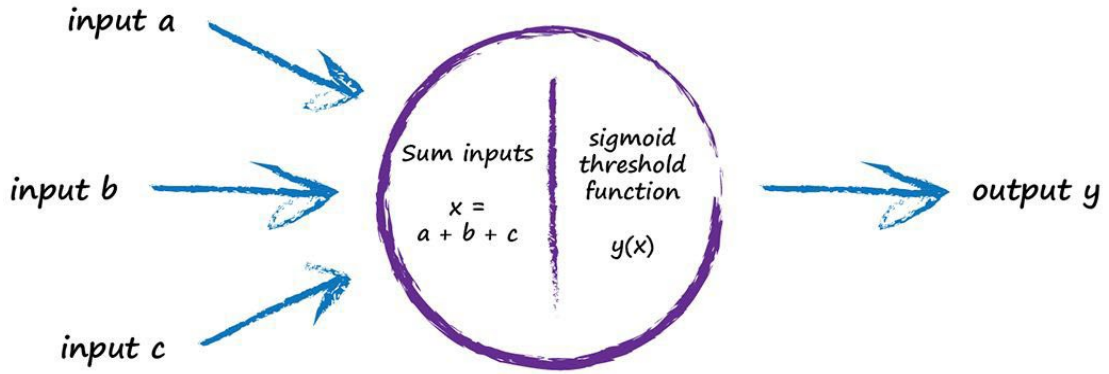


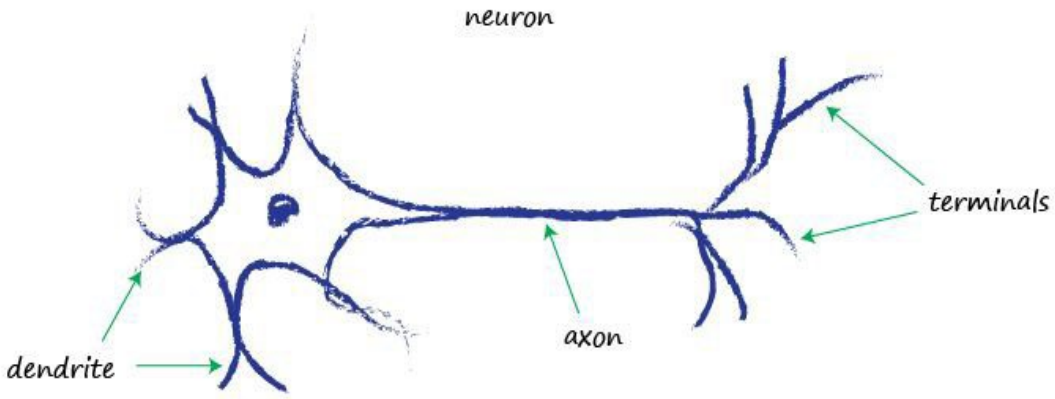
The Concept of Input / Output model



Artificial Neuron Model: input vs output



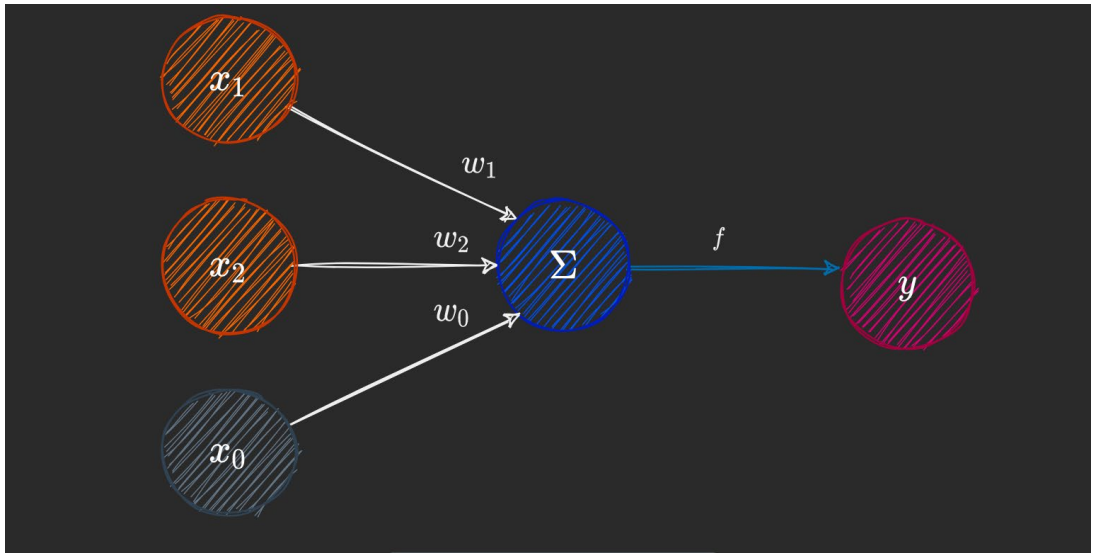




Artificial Neuron Model: PERCEPTRON structure

Perceptron Components :

- Input nodes x0 … Xn
- Output node y
- An activation function
- Weights and biases
- Summation
- Activation Function f



Artificial Neuron Model: PERCEPTRON maths

FEED FORWARD CACULATION:

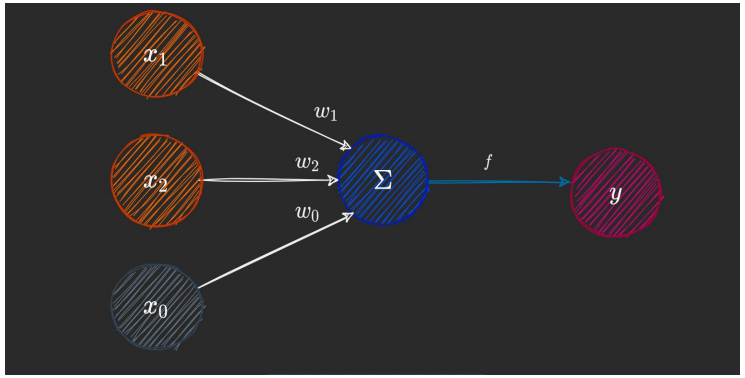
The output calculation is straightforward.

Compute the DOT PRODUCT of the input [x1 x2] and weight vector [w1 w2]

The bias b equals w0

Apply the activation function.

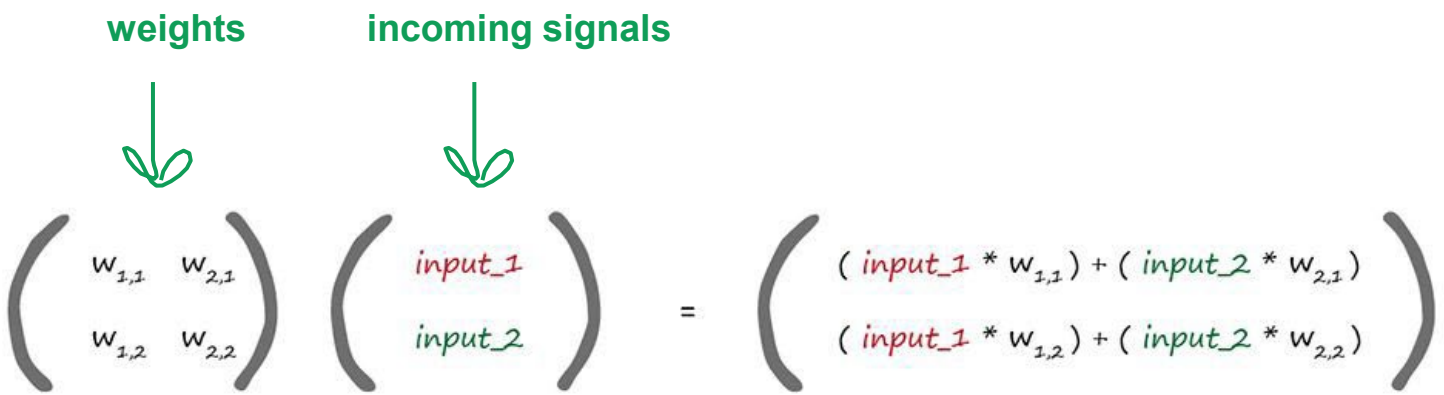
$$y = f(w \cdot X + b)$$



Feedforward computation:
(DOT PRODUCT is a form of mathematical summation)

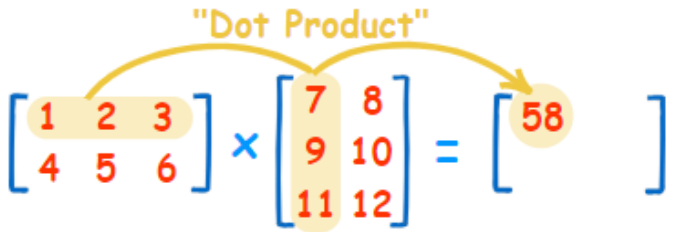
$$y = f(x_1 \cdot w_1 + x_2 \cdot w_2 + b)$$

Artificial Neuron Model:
PERCEPTRON maths



$$W \cdot I = X$$

dot product

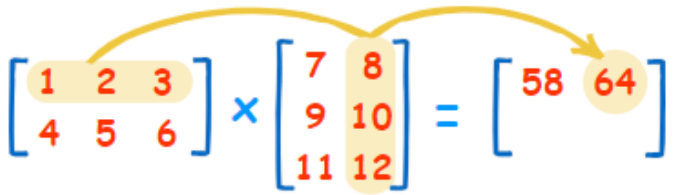


The "Dot Product" is where we **multiply matching members**, then sum up:

$$(1, 2, 3) \bullet (7, 9, 11) = 1\times7 + 2\times9 + 3\times11$$
$$= 58$$

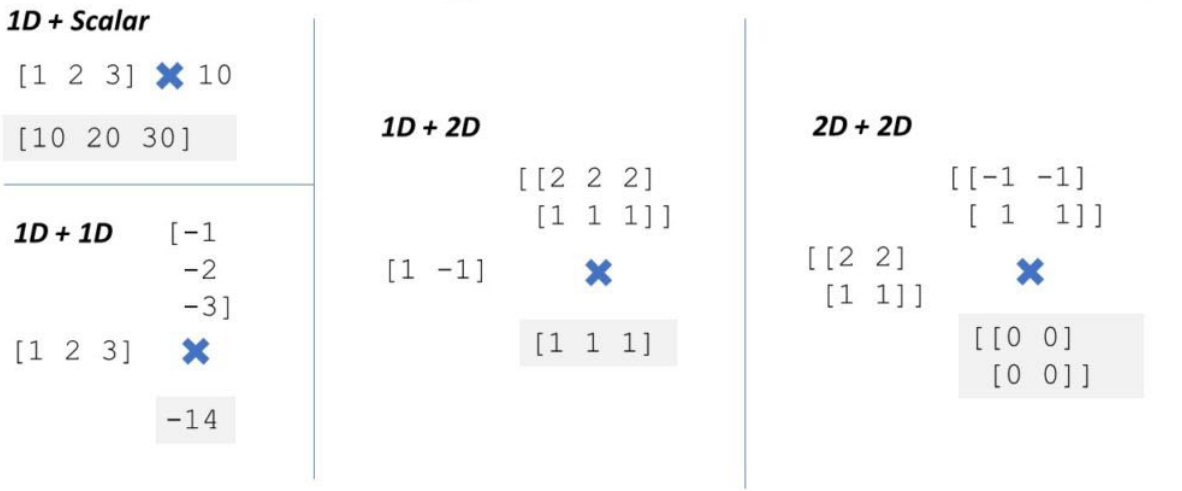
We match the 1st members (1 and 7), multiply them, likewise for the 2nd members (2 and 9) and the 3rd members (3 and 11), and finally sum them up.

Want to see another example? Here it is for the 1st row and **2nd column**:



$$(1, 2, 3) \bullet (8, 10, 12) = 1\times8 + 2\times10 + 3\times12$$
$$= 64$$

np.dot()



$$a \cdot b = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}_{(1 \times n)} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}_{(n \times 1)} = \left\{ a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5 \right\}$$

Dot Product

```
a = np.array([3, 5, 6])
b = np.array([23, 15, 1])

np.dot(a, b)
```

```
150
```

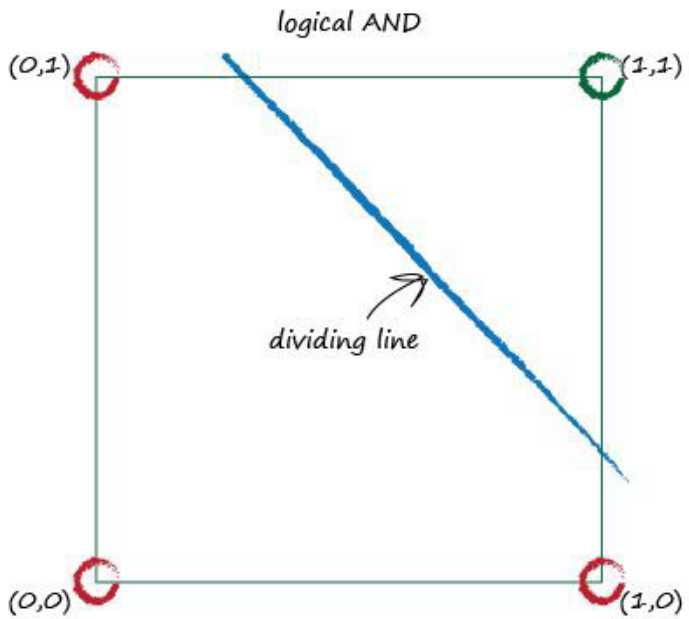
Boolean Logic

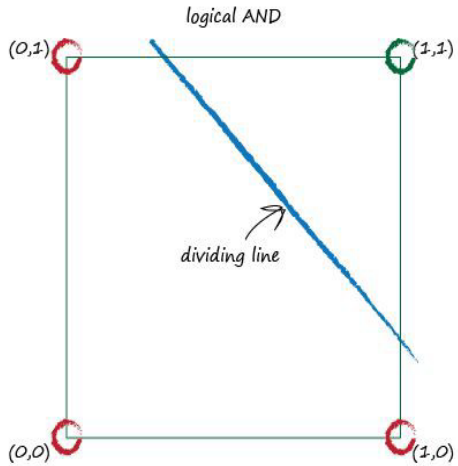
IF I have eaten my vegetables **AND** I am still hungry **THEN** I can have ice cream.

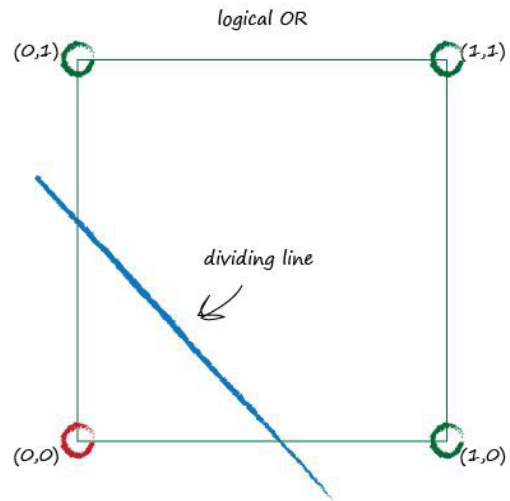
IF it's the weekend **OR** I am on annual leave **THEN** I'll go to the park.

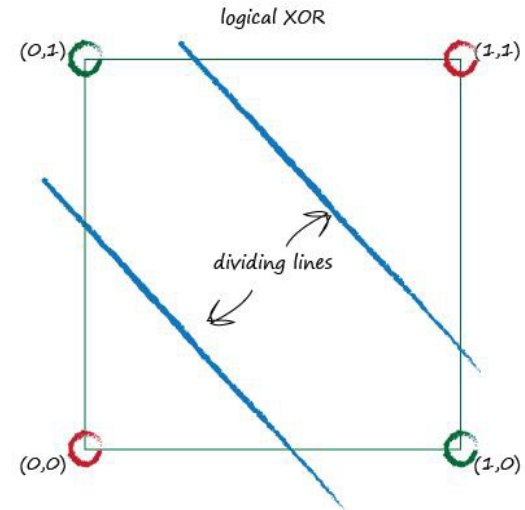
Input A	Input B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

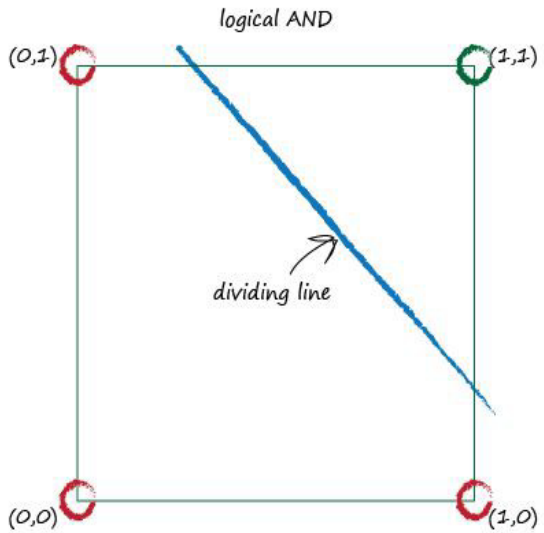
Input x1	Input x2	AND y
0	0	0
0	1	0
1	0	0
1	1	1

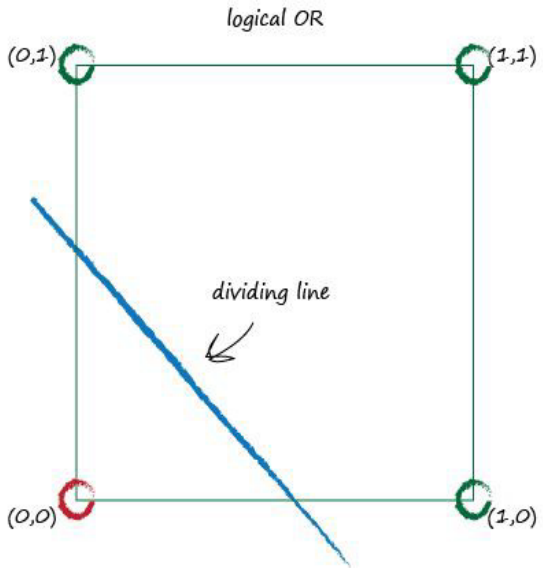



logical AND
(0,1)
(1,1)
dividing line
(0,0)
(1,0)


logical OR
(0,1)
(1,1)
dividing line
(0,0)
(1,0)


logical XOR
(0,1)
(1,1)
dividing lines
(0,0)
(1,0)



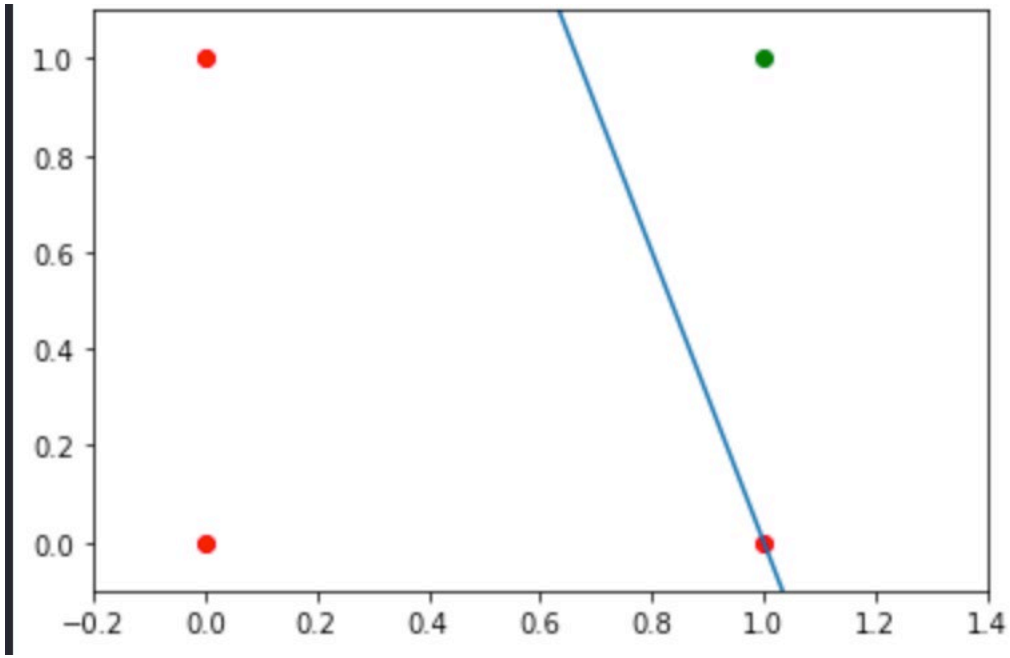


$$Class_1 : w \cdot X + b \geq 0$$

$$Class_2 : w \cdot X + b < 0$$

{01}
Fundamentals

```
p = Perceptron(weights=[0.3, 0.3, 0.3],
               learning_rate=0.2)

for in_data, label in labelled_samples(30):
    p.adjust(label,
             in_data)

test_data, test_labels = list(zip(*labelled_samples(30)))

evaluation = p.evaluate(test_data, test_labels)
print(evaluation)


fig, ax = plt.subplots()
xmin, xmax = -0.2, 1.4
X = np.arange(xmin, xmax, 0.1)
ax.scatter(0, 0, color="r")
ax.scatter(0, 1, color="r")
ax.scatter(1, 0, color="r")
ax.scatter(1, 1, color="g")
ax.set_xlim([xmin, xmax])
ax.set_ylim([-0.1, 1.1])
m = -p.weights[0] / p.weights[1]
c = -p.weights[2] / p.weights[1]
print(m, c)
ax.plot(X, m * X + c )
plt.plot()
```



PYTHON PERCEPTRON MODEL code

```
import Numpy as np
import Matlibplot.pyplot as plt
```

Learn how to specify: input/output relationship

Initialize weights

Forward computation

```python
import numpy as np
import matplotlib.pyplot as plt

class Perceptron(object):

    def __init__(self, no_of_inputs, no_of_training_epochs=20, learning_rate=0.01):
        self.no_of_training_epochs = no_of_training_epochs
        self.learning_rate = learning_rate
        self.weights = np.random.normal(0, 10, no_of_inputs + 1)
        self.bias = self.weights[0]
        self.epoch_list=[]
        self.error_history=[]


    def step_function(self, inputs):
        netto_summation = np.dot(inputs, self.weights[1:]) + self.bias
        if netto_summation > 0:
          activation = 1
        else:
          activation = 0
        return activation

    def train(self, training_inputs, teacher_labels):
        epoch =0;
        i =0;
        for _ in range(self.no_of_training_epochs):
          epoch+=1
          for inputs, teacher in zip(training_inputs, teacher_labels):
              activated_perceptron_output = self.step_function(inputs)
              error = (teacher - activated_perceptron_output)
              self.weights[1:] += self.learning_rate * (error) * inputs
              self.bias += self.learning_rate  * error
              i+=1
              self.epoch_list.append(epoch)
              self.error_history.append(np.sum(np.abs(error)))
```

PYTHON PERCEPTRON MODEL code

import Numpy as np
import Matlibplot.pyplot as plt

Where are input & ouput specified?

Where are the weights initialized?

How is the Forward computation specified?

To Do:
Change the code as to print
Interation number
Bach number
Error
And weights

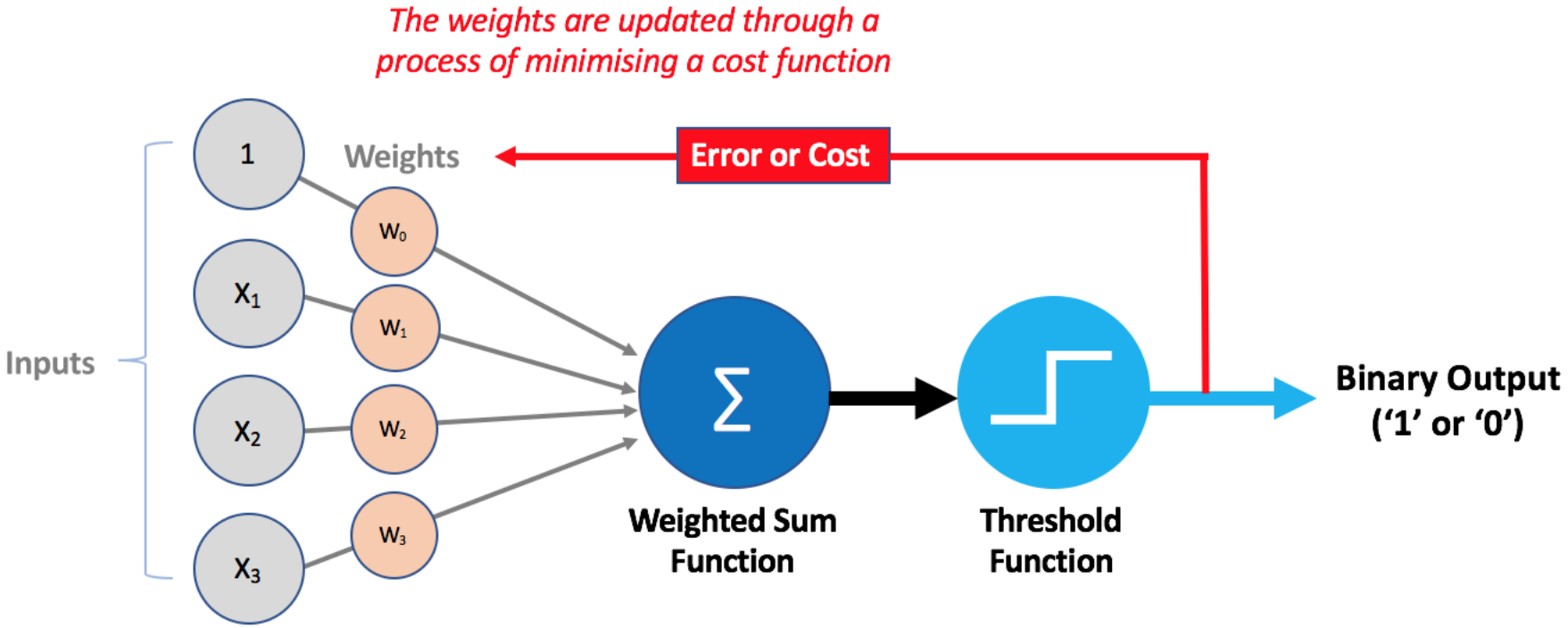
PYTHON
INPUT / OUTPUT relationship code

```
import Numpy as np
import Matlibplot.pyplot as plt

import numpy as np

inputs = []
inputs.append(np.array([1, 1]))
inputs.append(np.array([1, 0]))
inputs.append(np.array([0, 1]))
inputs.append(np.array([0, 0]))

teacher = np.array([1, 0, 0, 0])
```

USING ZIP

```
print([inputs for teacher in zip(inputs, teacher)])

print([i for i in zip(inputs, teacher)])
```

NEXT WEEK:


The weights are updated through a process of minimising a cost function
Error or Cost
Weights
1
X_1
X_2
X_3
Inputs
W_0
W_1
W_2
W_3
Σ
Weighted Sum Function
Threshold Function
Binary Output ('1' or '0')

Artificial Neuron Model:
PERCEPTRON maths

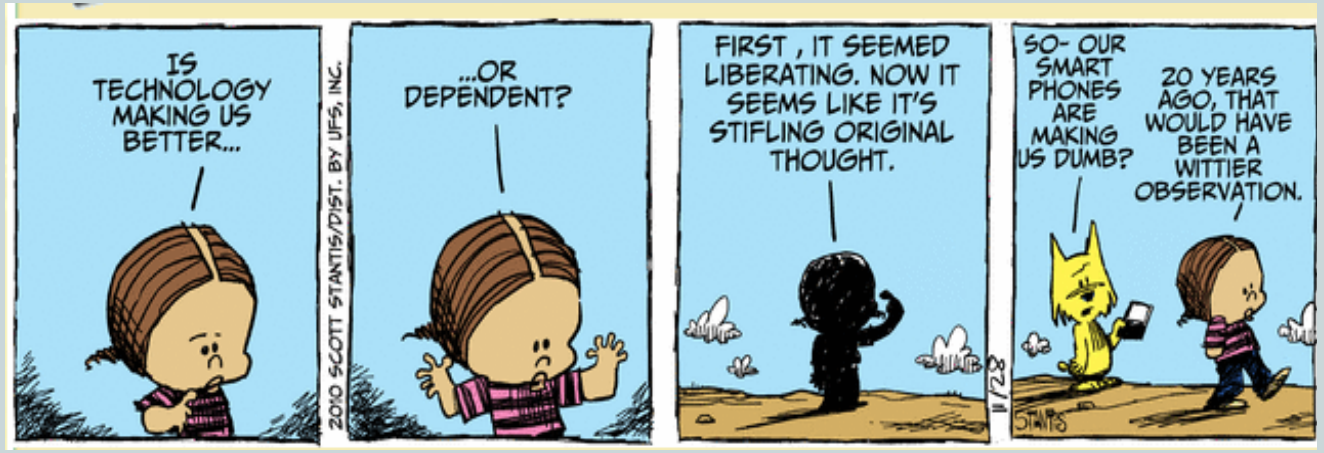
$$\Delta w = node \cdot (actual - computed)$$

Error calculation

Artificial Neuron Model:
PERCEPTRON maths

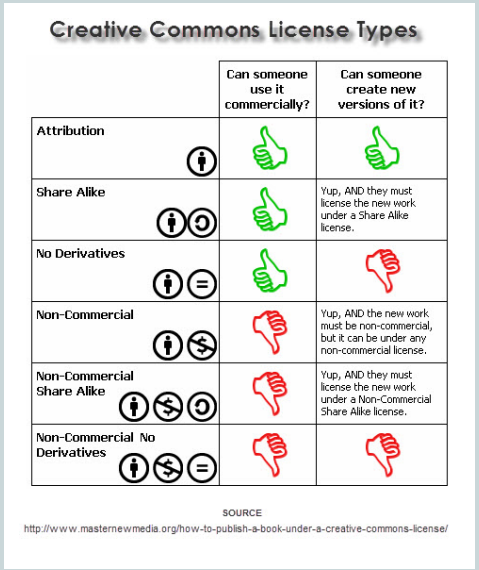
$$w_{updated} = w_{old} + lr \cdot \Delta w$$

FEED BACKWARD calculation



This lesson was developed by:

Robert Frans van der Willigen
CMD, Hogeschool Rotterdam
OKT 2020



http://empoweringthenatives.edublogs.org/2012/03/15/creative-commons-licenses/

http://creativecommons.org/licenses/


the functional art
alberto cairo
TECHNOLOGY TRENDS TO WATCH
Christian Leborg
Visual Grammar
the analytics revolution
Bill Franks
MADE to STICK
generative art
matt pearson
EMPATHIC THINGS
INTERNET OF THINGS
MAKERS
THE NEW INDUSTRIAL REVOLUTION
CHRIS ANDERSON
MARITIEME MAAKINDUSTRIE TERUG NAAR RDM
TV AND VIDEO
An analysis of evolving consumer habits
THE DISRUPTION MACHINE
Learning to Program with MATLAB
BUILDING GUI TOOLS
Port Management Program
R Data Analysis
10 TRENDS FOR 2014
A giant cage
resonate
Design for Information
HACKATHO
OPEN HAVE
1 VOLLE DAG
7 SEPTEMBER
WEREldHAVENDAGEN 2013
the myths of innovation
scott berkun
O'REILLY
Cool Infograph
Effective Communication with Data Visualization and Design
The VISU
PHIL SIMON
INVENT TO LEARN
Making, Tinkering, and Engineering in the Classroom
MORE SECRETS with Big Data Analytics
Processing
A Programming Handbook for Visual Designers and Artists
OUR WORK HERE IS DONE
VISIONS OF A ROBOT ECONOMY
THE WALL STREET JOURNAL.
Guide to Information Graphics
Dona M. Wong
Customer Journey Mapping
Wearable Technology Design
The Hidden Pitfall in your Design
Making Things See
Learn by Discovery
TEACHING that STICKS
Chip Heath & Dan Heath
Protecting Data and Privacy in the Cloud
SOGETI
DISCOVER METEOR
Making Things Talk
The Internet of Things
Business Adapts to a New Style of Computer
EVOLUTION OF AGILE