Neural Networks (1/2)

Lecture 7

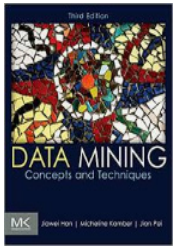
IDS-L7

Chair of Process and Data Science

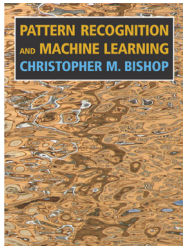
RWTH AACHEN UNIVERSITY

Outline of Today's Lecture

- Introduction to Neural Networks
- Human and Artificial Neurons
- What are Neural Networks?
- Model Representation
- Feedforward Networks

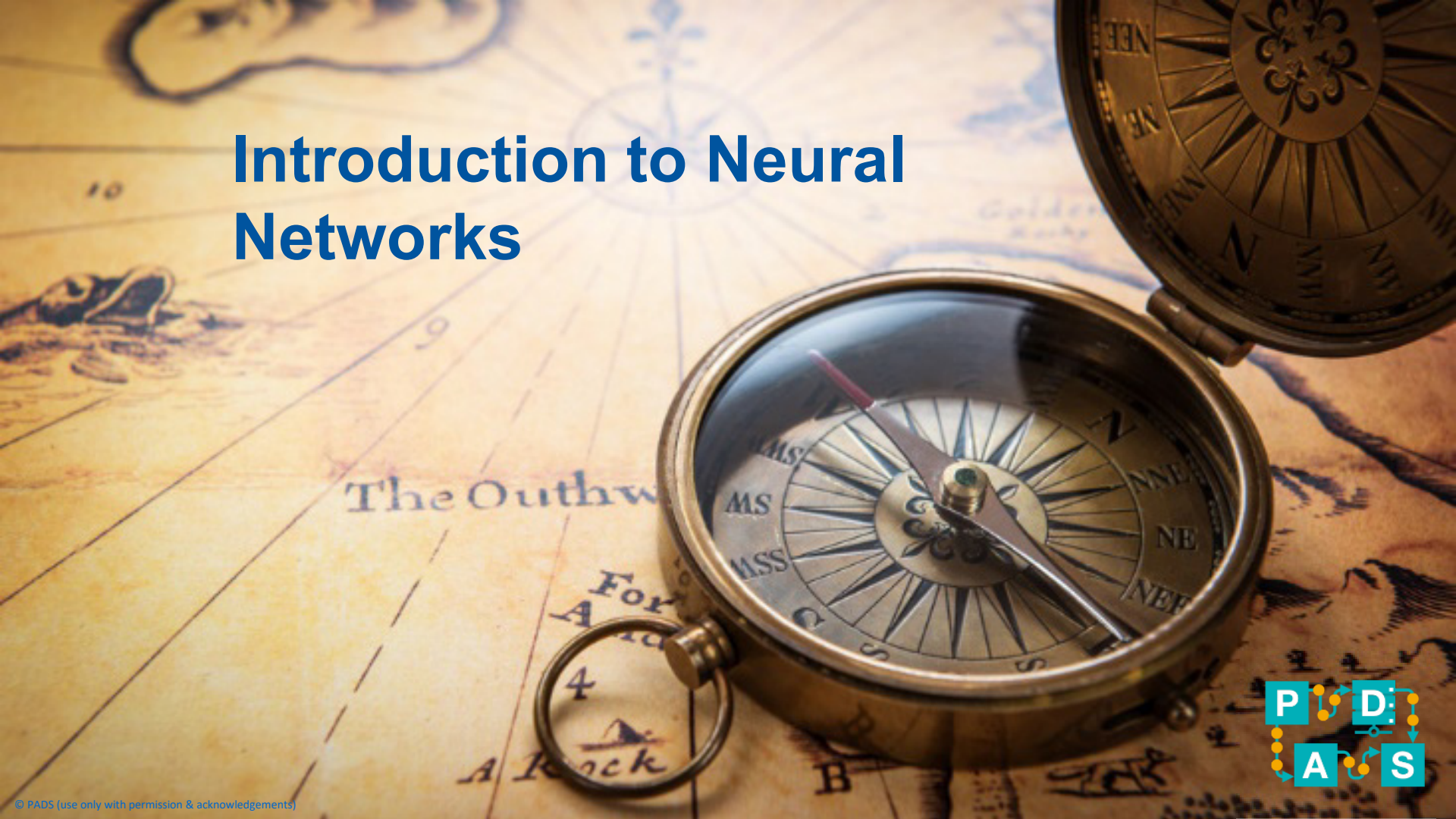


Based on chapter 9 of "data mining concepts and techniques" by Jiawei Han



Chapter 5 of "pattern recognition and machine learning" by Christopher M. Bishop

Introduction to Neural Networks



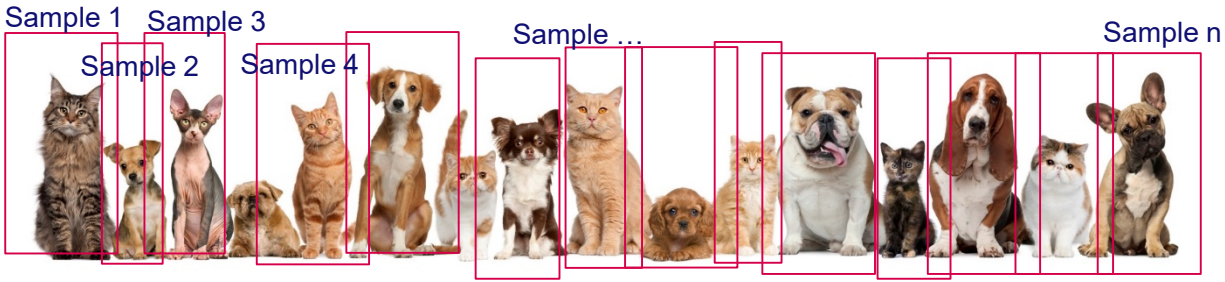
Positioning Neural Networks

Supervised learning techniques

- **Decision Trees**
  - **Were initially developed for categorical features and then extended to continuous features.**
- **Regression**
  - **Followed the reverse path (most suitable for continuous data).**
- **Support Vector Machines**
  - **For large numbers of features while avoiding overfitting.**
- **Today: Neural Networks**
  - **Flexible in the types of data they can support.**
  - **Learns the important features from basically any data structure.**

Why Use Neural Networks?

Linear machines are often not satisfactory when facing real world situations



Is this a dog?

A five years old child simply knows, but how should a computer know?

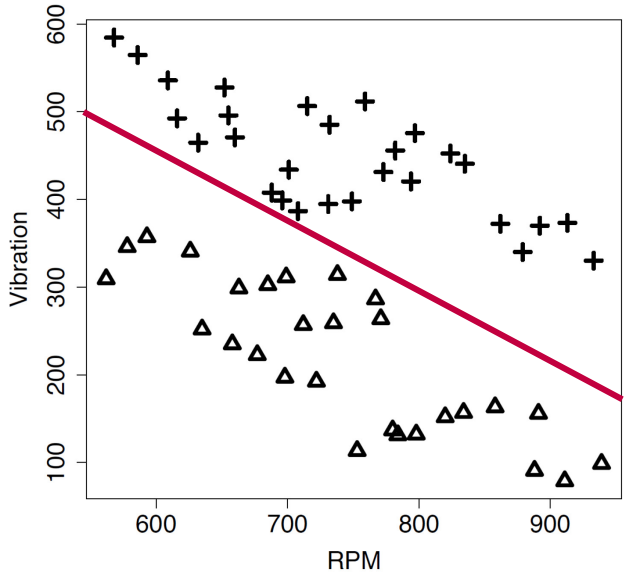
Chair of Process and Data Science

Why Use Neural Networks?
Linear Problem

For simple linear classification: one can use logistic regression or SVM to separate the instances

ID	RPM	VIBRATION	STATUS
1	568	585	good
2	586	565	good
3	609	536	good
4	616	492	good
5	632	465	good
6	652	528	good
7	655	496	good
8	660	471	good
9	688	408	good
10	696	399	good
11	708	387	good
12	701	434	good
13	715	506	good
14	732	485	good
15	731	395	good
16	749	398	good
17	759	512	good
18	773	431	good
19	782	456	good
20	797	476	good
21	794	421	good
22	824	452	good
23	835	441	good
24	862	372	good
25	879	340	good
26	892	370	good
27	913	373	good
28	933	330	good

ID	RPM	VIBRATION	STATUS
29	562	309	faulty
30	578	346	faulty
31	593	357	faulty
32	626	341	faulty
33	635	252	faulty
34	658	235	faulty
35	663	299	faulty
36	677	223	faulty
37	685	303	faulty
38	698	197	faulty
39	699	311	faulty
40	712	257	faulty
41	722	193	faulty
42	735	259	faulty
43	738	314	faulty
44	753	113	faulty
45	767	286	faulty
46	771	264	faulty
47	780	137	faulty
48	784	131	faulty
49	798	132	faulty
50	820	152	faulty
51	834	157	faulty
52	858	163	faulty
53	888	91	faulty
54	891	156	faulty
55	911	79	faulty
56	939	99	faulty



Why Use Neural Networks?
Non-Linear Problem

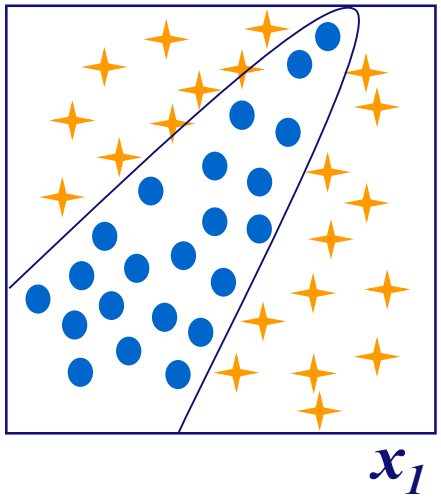
- **Nonlinear classification may work fine.**

$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

from lecture on regression

$$\min_{\vec{w},b,\epsilon} \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m} \epsilon_i$$
such that
$$y_i(\vec{w} \cdot \boldsymbol{\phi}(\vec{x}_i) + b) \geq 1 - \epsilon_i \text{ for any } i$$

from lecture on SVMs



- **But for most of the problems in practice:**
  - The number of features is often more than two.
  - What if the number of features is 100?
  - Just the number of possible pairs is already $n(n-1) = 9900$!

Why Neural Networks?
Limitation of Simple Linear Models

- **Consider the following as a general form of Linear Models:**
- **Where:**
  - x respresents input (vector describing the descriptive features)
  - w represents the weight vector
  - $\phi_j(x)$ is:
    - jth element of the feature vector
    - or a calculation to change input dimensions to a higher dimension
  - f *is*:
    - a simple threshold-based function returning *0 or 1*
    - or more sophisticated functions like the sigmoid function (to add nonlinearity)
  - M represents the number of descriptive features

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \boldsymbol{\phi}_j(\boldsymbol{x}) \right)$$

Why Neural Networks?
Limitation of Simple Linear Models

- **SVM and Neural Networks try to address limitations of traditional purely linear approaches**
  - **SVM**
    - **By varying number of $\phi_j(x)$ functions on training data points**
  - **Neural Networks**
    - **$\phi_j(x)$ functions may be based on other layers and can be learned**

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}) \right)$$

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

- input x has D dimensions (features)
- σ and h are activation functions
  (will be explained later)

Why Use Neural Networks?

Example

- Automatic dog detection
- Provide many labeled sample of
  - Dogs
  - Non dogs

Non dogs

Dogs

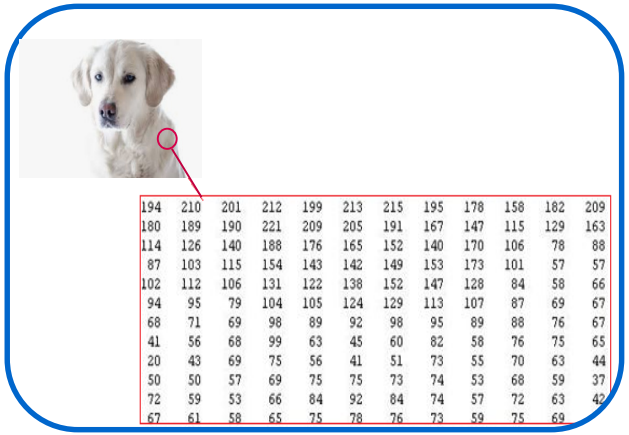
- Is this a dog?

Chair of Process and Data Science

Why Use Neural Networks?

Large number of features

What we see!



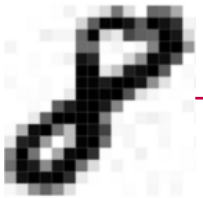
What the camera sees!

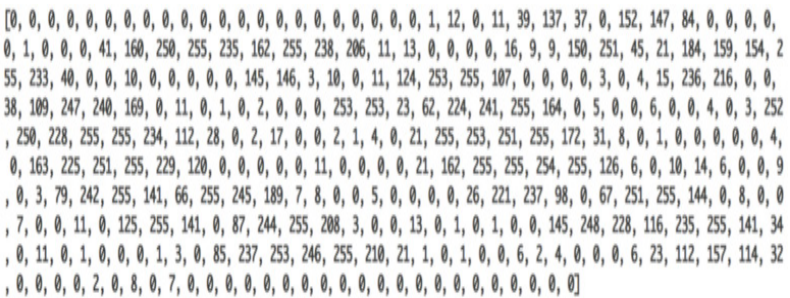
Consider the number of features!

More Specific Example
Number recognition

- **To a computer, an image is really just a grid of numbers that represent how dark each pixel is:**
  - **Now imagine: handwriting of digit number 8**
    - 18*18 pixel image as an array of 324 numbers

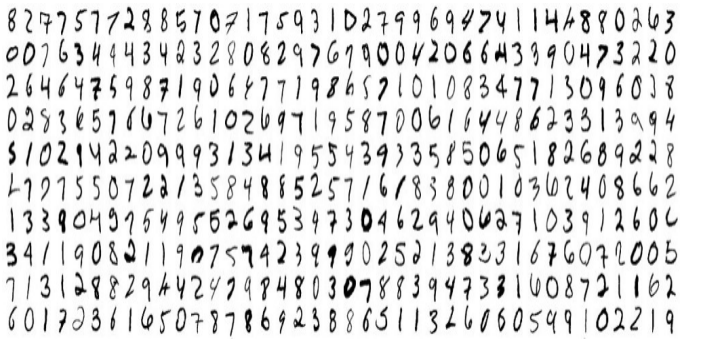
The numbers inside pixels show the darkness of that pixel.
In grey scale format ,the value of each pixel is between 0-255





[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 12, 0, 11, 39, 137, 37, 0, 152, 147, 84, 0, 0, 0, 0,
0, 1, 0, 0, 0, 41, 160, 250, 255, 235, 162, 255, 238, 206, 11, 13, 0, 0, 0, 0, 16, 9, 9, 150, 251, 45, 21, 184, 159, 154, 2
55, 233, 40, 0, 0, 10, 0, 0, 0, 0, 0, 145, 146, 3, 10, 0, 11, 124, 253, 255, 107, 0, 0, 0, 0, 3, 0, 4, 15, 236, 216, 0, 0,
38, 109, 247, 240, 169, 0, 11, 0, 1, 0, 2, 0, 0, 0, 253, 253, 23, 62, 224, 241, 255, 164, 0, 5, 0, 0, 6, 0, 0, 4, 0, 3, 252
, 250, 228, 255, 255, 234, 112, 28, 0, 2, 17, 0, 0, 2, 1, 4, 0, 21, 255, 253, 251, 255, 172, 31, 8, 0, 1, 0, 0, 0, 0, 4,
0, 163, 225, 251, 255, 229, 120, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 21, 162, 255, 255, 254, 255, 126, 6, 0, 10, 14, 6, 0, 0, 9
, 0, 3, 79, 242, 255, 141, 66, 255, 245, 189, 7, 8, 0, 0, 5, 0, 0, 0, 0, 26, 221, 237, 98, 0, 67, 251, 255, 144, 0, 8, 0, 0
, 7, 0, 0, 11, 0, 125, 255, 141, 0, 87, 244, 255, 208, 3, 0, 0, 13, 0, 1, 0, 1, 0, 0, 145, 248, 228, 116, 235, 255, 141, 34
, 0, 11, 0, 1, 0, 0, 0, 1, 3, 0, 85, 237, 253, 246, 255, 210, 21, 1, 0, 1, 0, 0, 6, 2, 4, 0, 0, 0, 6, 23, 112, 157, 114, 32
, 0, 0, 0, 0, 2, 0, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

How computer sees this 8
(Set of features)



Training dataset for the number recognition classifier

https://medium.com

Chair of Process and Data Science

What Is a Neural Network?

- *Motivation*
  - Simulation of the neuron system's information processing mechanisms (as used in the human brain).
- *Structure*
  - Huge amount of densely connected, mutally operating processing units (*neurons*)
- It learns from experiences (training instances)

What Is a Neural Network?

- An information processing paradigm
- Inspired by biological nervous systems
  - such as the human brain …
- Composed of a large number of highly interconnected processing elements (neurons)
- Through a learning process
  - Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

Historical Background

- **First artificial neuron**
  - **1943**
  - **Invented by the neurophysiologist Warren McCulloch and the logician Walter Pits**

- **But the technology available at that time did not allow them to use such artificial neurons on any meaningful manner.**

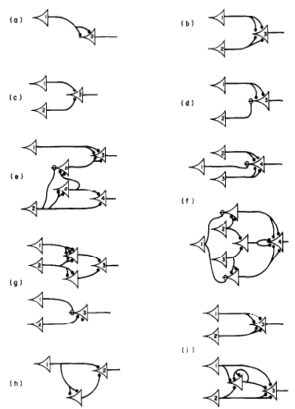
A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

WARREN S. MCCULLOCH AND WALTER PITTS
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

1. Introduction. Theoretical neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from $<1\ \mathrm{ms}^{-1}$ in thin axons, which are usually short, to $>150\ \mathrm{ms}^{-1}$ in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time of arrival of impulses at points unequally remote from the same source. Excitation across synapses occurs predominantly from axonal terminations to somata. It is still a moot point whether this depends upon irreciprocity of individual synapses or merely upon prevalent anatomical configurations. To suppose the latter requires no hypothesis *ad hoc* and explains known exceptions, but any assumption as to cause is compatible with the calculus to come. No case is known in which excitation through a single synapse has elicited a nervous impulse in any neuron, whereas any neuron may be excited by impulses arriving at a sufficient number of neighboring synapses within the period of latent addition, which lasts <0.25 ms. Observed temporal summation of impulses at greater intervals

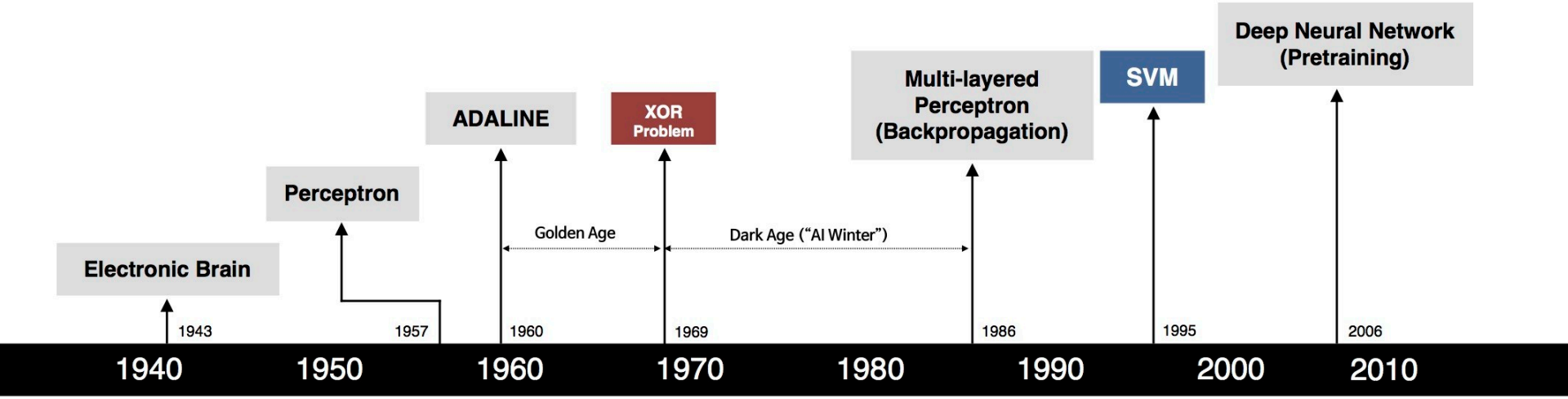
* Reprinted from the *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133 (1943).



Chair of Process and Data Science

Brief History of Neural Networks

(taken from Sefik Ilkin Serengil's Blog on the "Evolution of Neural Networks" Oct. 2017)



S. McCulloch - W. Pitts

F. Rosenblatt

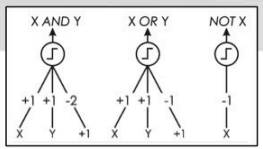
B. Widrow - M. Hoff

M. Minsky - S. Papert

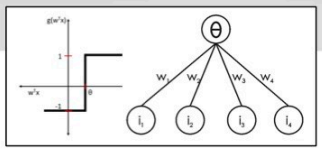
D. Rumelhart - G. Hinton - R. Wiliams

V. Vapnik - C. Cortes

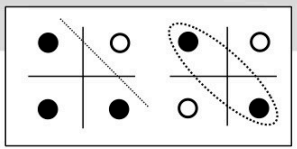
G. Hinton - S. Ruslan



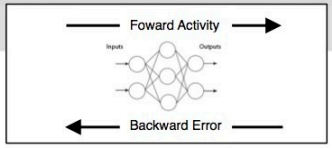
- Adjustable Weights
- Weights are not Learned



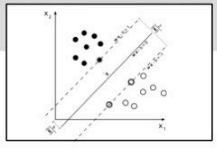
- Learnable Weights and Threshold



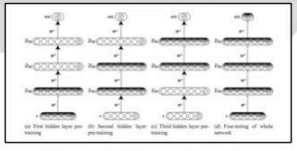
- XOR Problem



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



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



- Hierarchical feature Learning

Why Use Neural Networks?

Why	• Ability to derive meaning
	  • Complicated data
	  • Imprecise data

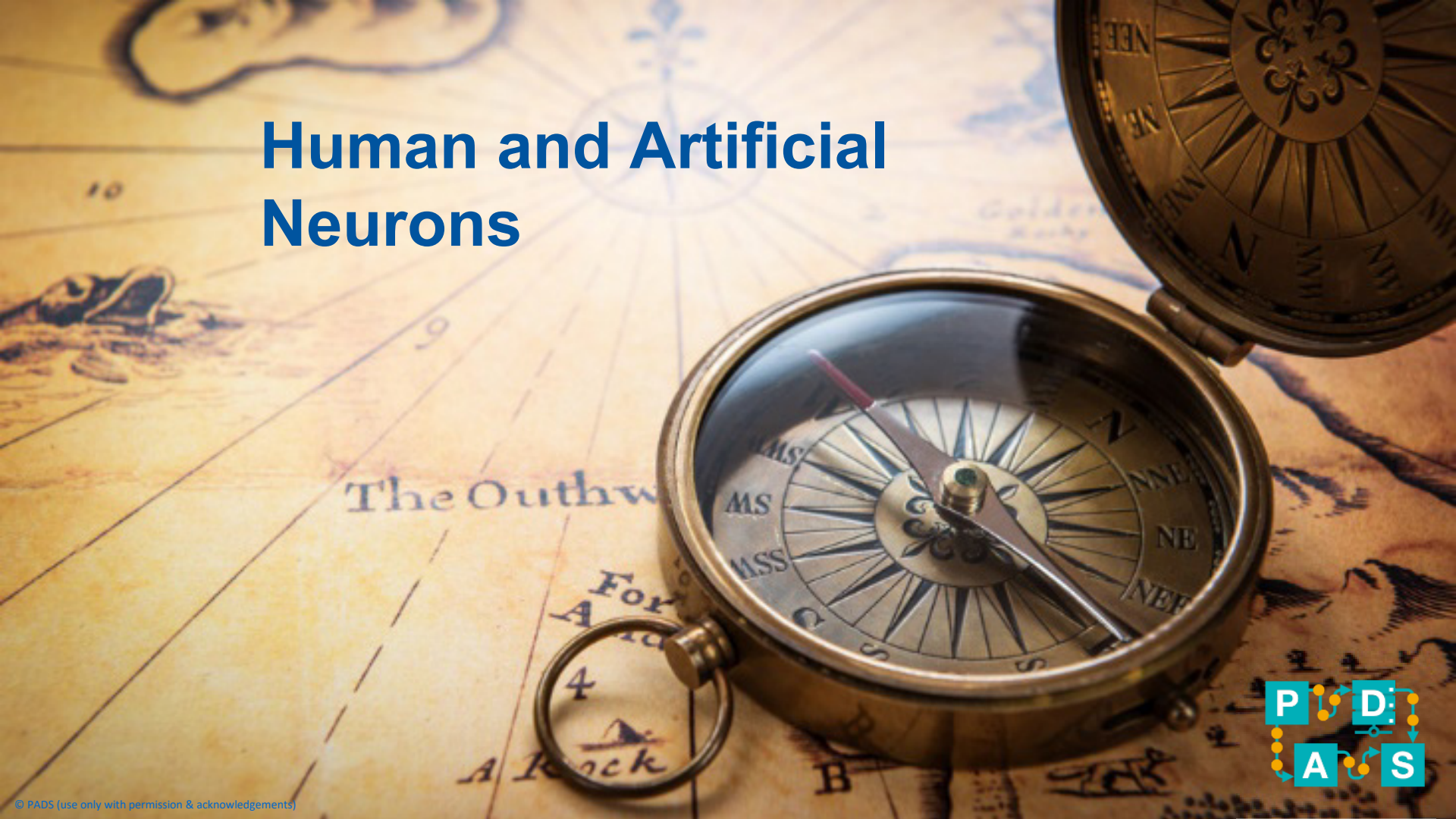
Uses	• Extract pattern • Detect trends that: • Humans and other computer techniques can’t
Advantages	• Adaptive learning • Self-Organization • Real Time Operation • Fault Tolerance

A trained neural network	• Is an expert that:
	  • can be used to provide projections given:
	  • New situations of interest
	  • Answer "what if" question

Properties of Artificial Neural Networks

- High level abstraction of neural input-output transformation
  - Inputs → weighted sum of inputs → nonlinear function → output
    - Typically no spikes (neurons fire in each cycle)
    - Typically use additional constraints or learning rules
- Often used where data or functions are uncertain
  - Goal is to learn from a set of training data
  - And to generalize from learned instances to new unseen data

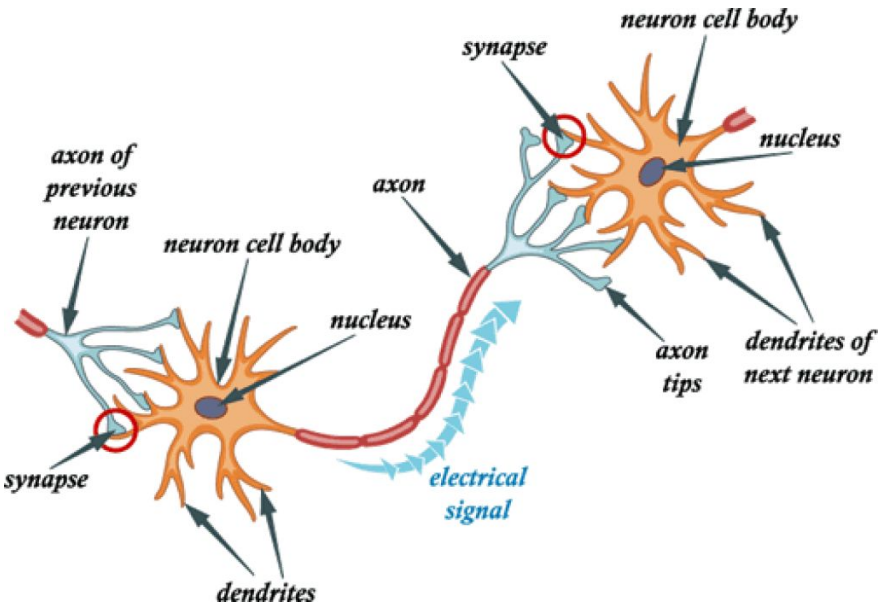
Human and Artificial Neurons



The human brain contains around 100 billion neurons (give or take a few billion).

How the Human Brain Learns

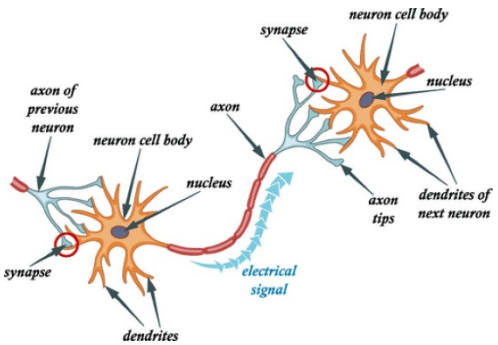
- Dendrites are sensitive to a certain kind of chemicals
- When there are a certain variety of chemical signals around these dendrites, they absorb them
  - When a certain amount of chemical is accumulated at the cell, it fires.
  - It generates an electrical pulse through an axon, into the synapse connected to the next neuron's dendrites.
  - The pulse is picked up as a signal by the dendrites of the next cell.

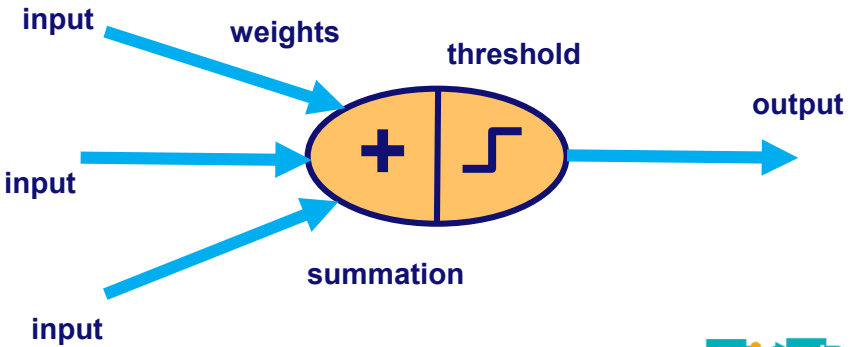


Chair of Process and Data Science

From Human Neurons to Artificial Neurons

- When a neuron receives excitatory input
  - If is sufficiently large compared with its inhibitory input, then sends a spike of electrical activity down its axon.
- Learning occurs:
  - By changing the effectiveness of the synapses.
  - The influence of one neuron on another changes.





Chair of Process and Data Science

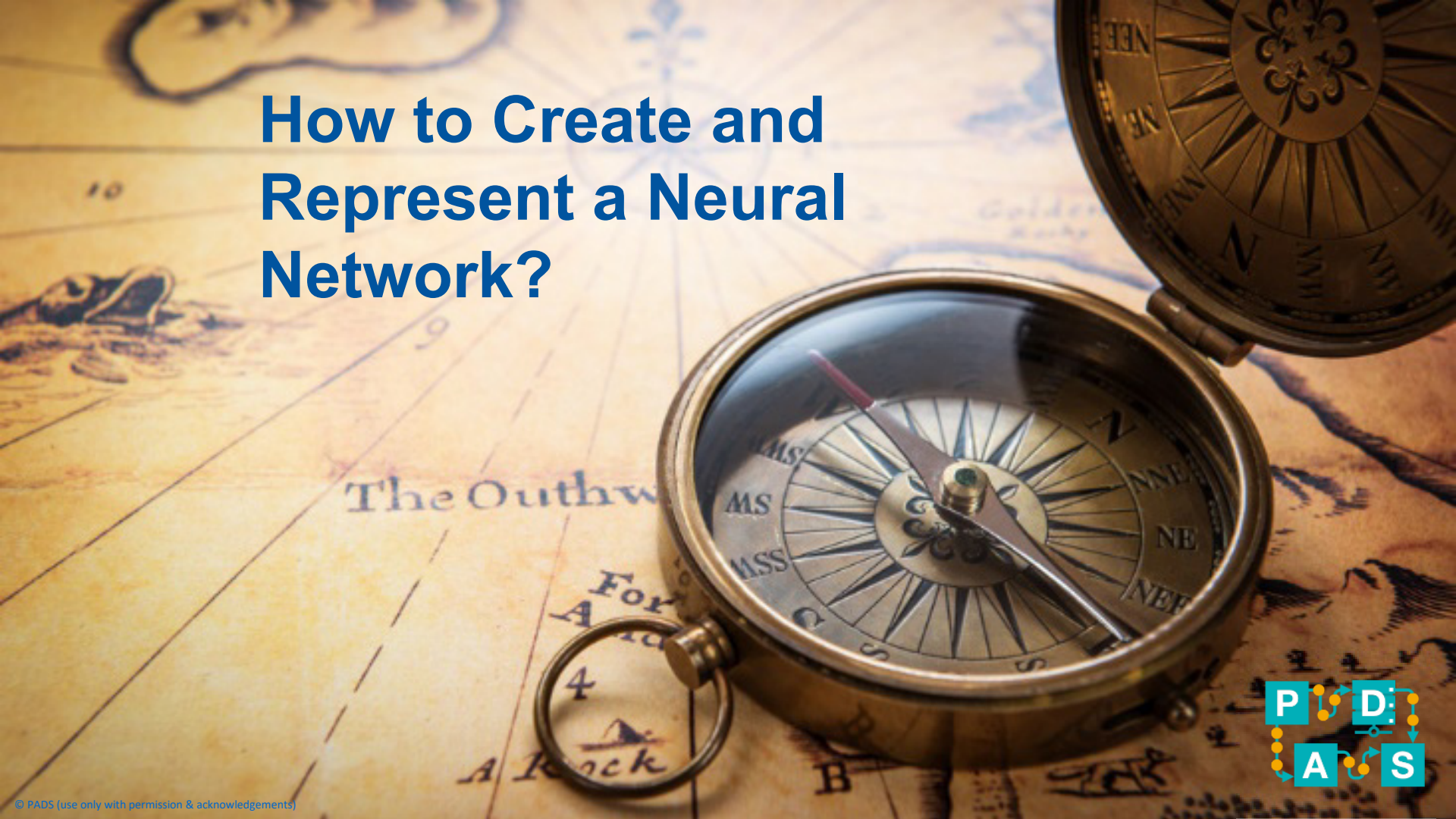
From Human Neurons to Artificial Neurons

- **“Imitating” neural networks**
  - **First**
    - **Deduce the essential features of neurons**
    - **Understand their interconnections**
  - **Then**
    - **Program a computer to simulate these features**

Our knowledge of neurons is incomplete and our computing power is limited.

Hence, our models are necessarily gross idealizations of real networks of neurons.

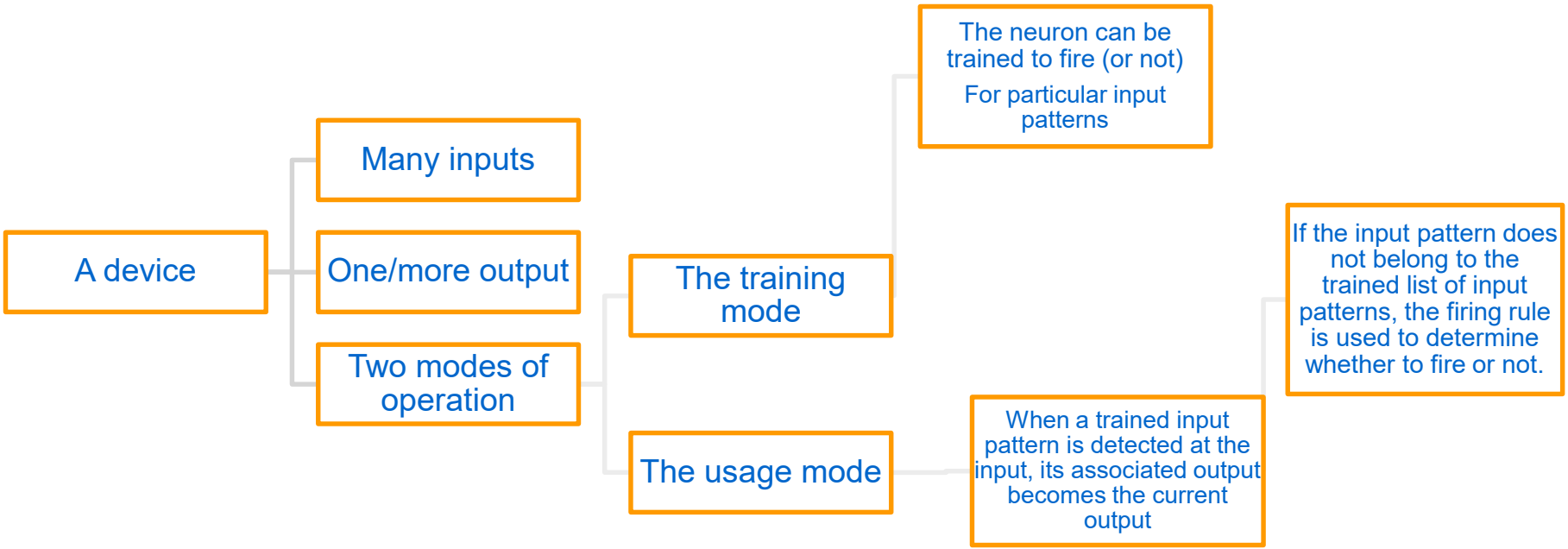
How to Create and Represent a Neural Network?



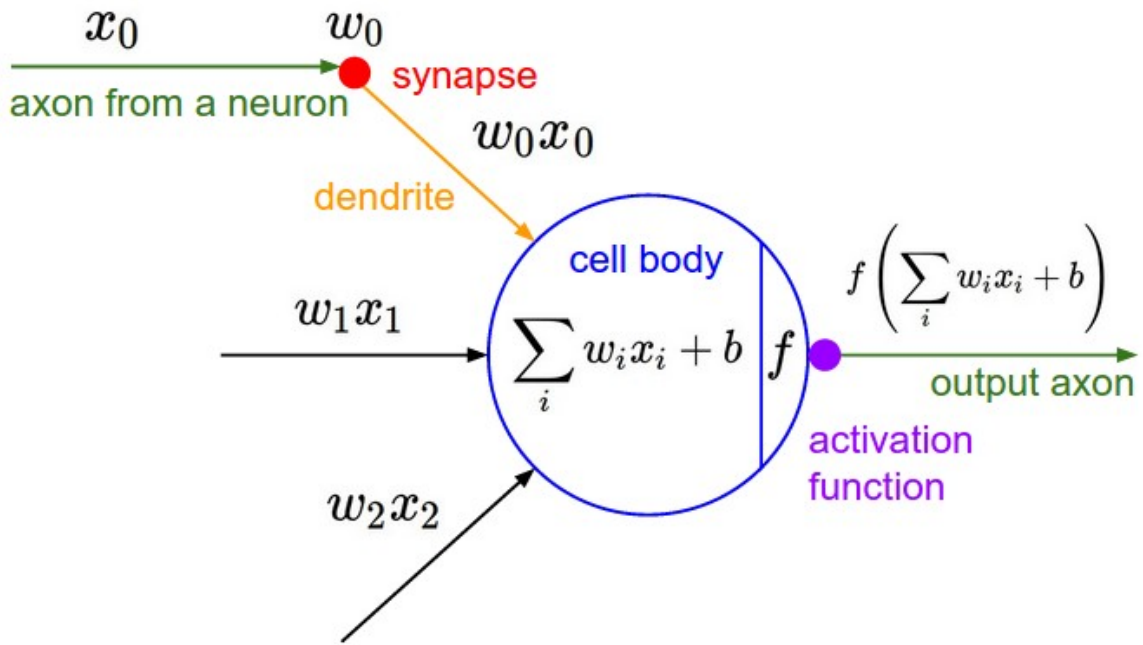
Create Neural Networks

- Neural Networks belong to the class of error-based learning algorithms (like SVMs, etc.)
- First
  - We learn how to create a network (focus of this lecture)
- Then
  - How to train the network using error (next lecture).

A Simple Neuron
Description of a simple neuron



A Simple Neuron



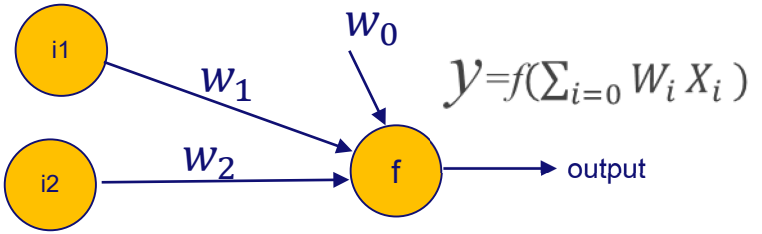
Chair of Process and Data Science

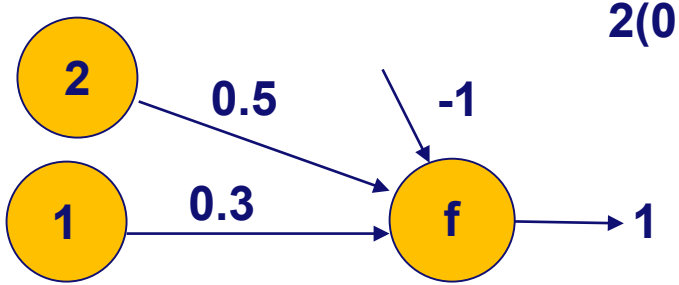
A Simple Neuron

- Every simple neuron acts as an actual brain neuron
  - get input signals
    - Inputs
  - assign importance to each input
    - Weights
  - analyze all the incoming signals with respect to their importance
    - Cell body (summation of weighted inputs)
  - use a threshold to decide: fire or not
    - Activation function

Simple Neuron Learning Process
Perceptron Example

- Randomly assign weights (between 0-1)
- Present inputs from training data
- Get output
  - Nudge weights to get results toward our desired target output
  - Repeat; stop when no errors, or enough epochs completed





2(0.5) + 1(0.3) + -1 = 0.3

output=1

assume threshold of 0

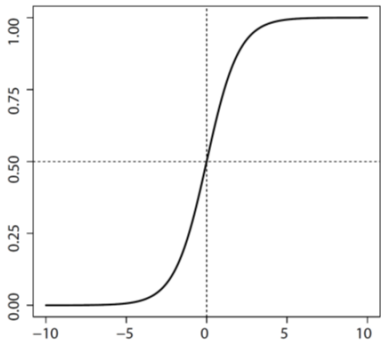
Chair of Process and Data Science

How to use a perceptron network?
Example

- Generally perceptron networks are used to learn how to make classifications
- For example
  - You have collected some data regarding the diagnosis of patients with heart disease
  - Age, Sex, Chest Pain Type, Resting BPS, Cholesterol, …, Diagnosis (0/1))
  - `67,1,4,120,229,…, 1`
  - `37,1,3,130,250,… ,0`
  - `41,0,2,130,204,… ,0`
- Train network to predict heart disease of new patients

Activation Function

- **Neurons use a threshold to decide when to fire**
  - **In neural networks the sigmoid function (like in logistic regression) is commonly used as an activation function:**



$$sigmoid = \frac{1}{1 + e^{-x}}$$

Remember: tends to go to 0 or 1.

- **However, now for simplicity we use step function:**

$$f(y) = \begin{cases} 0, & y < 0 \\ 1, & y \geq 0 \end{cases}$$

Chair of Process and Data Science

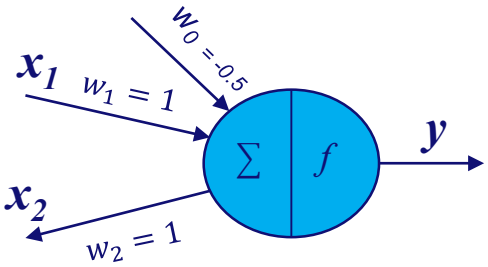
What is an artificial neuron doing when it learns?
Logical OR example

Logical OR function:

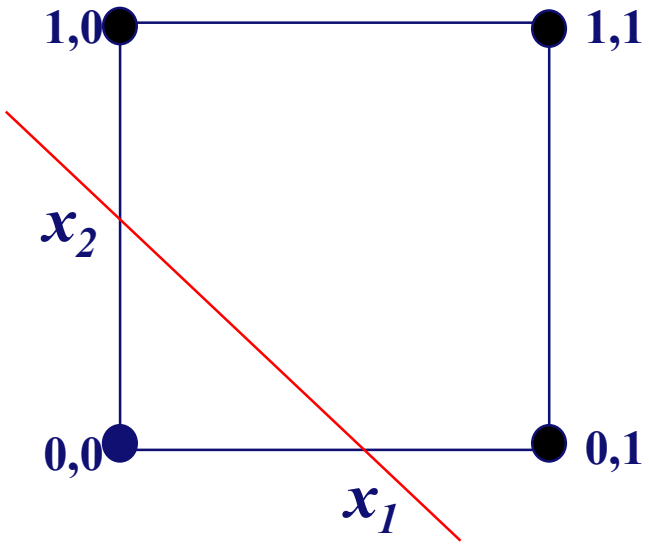
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

$y = f(x_1, x_2)$

Simple Neural Network:



$y = f(w_0 + w_1 x_1 + w_2 x_2)$



$$1 * x_1 + 1 * x_2 - 0.5 = y \;:\; f(y) = \begin{cases} 0, & y < 0 \\ 1, & y \geq 0 \end{cases}$$

Chair of Process and Data Science

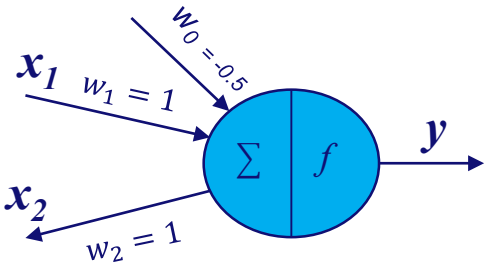
How neural networks are different?

Logical XOR example

Logical XOR function

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

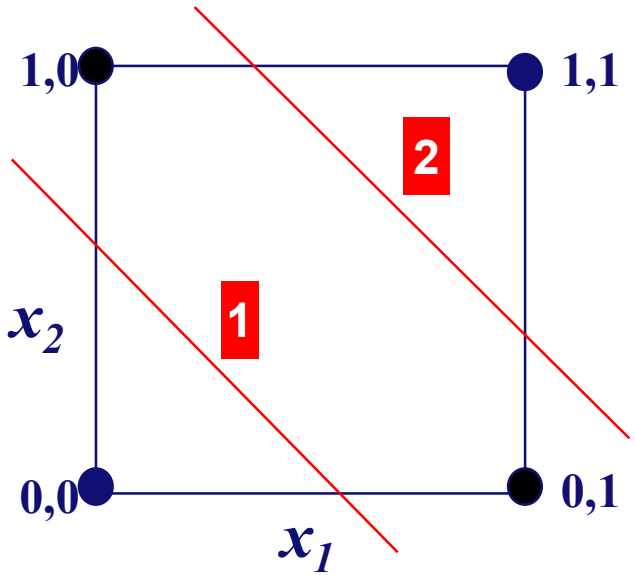
Simple Neural Network:



$$y = f(w_0 + w_1 x_1 + w_2 x_2)$$

1 $\begin{cases} w_0 = -0.5 \\ w_1 = 1 \\ w_2 = 1 \end{cases}$ $1 * x_1 + 1 * x_2 - 0.5 = sum$: $f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$

2 $\begin{cases} w_0 = -1.5 \\ w_1 = 1 \\ w_2 = 1 \end{cases}$ $1 * x_1 + 1 * x_2 - 1.5 = sum$: $f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$



The simple Neural Network cannot produce proper classifications!

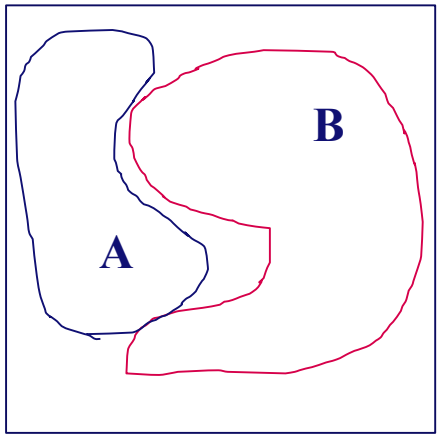
Chair of Process and Data Science

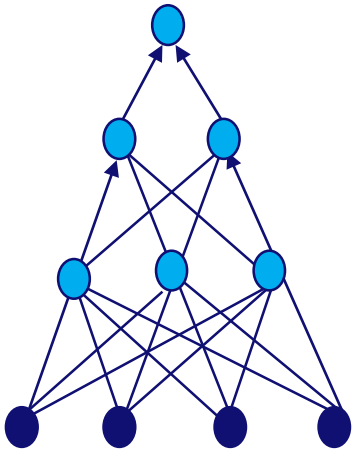
Limitations of Simple Neural Networks

- What is a Perceptron doing when it learns?
  - A discrimination function (separating plane) is generated
  - Has the power to map input patterns to output class values
- Limitations of the basic Perceptron
  - Able to form only linear discriminate functions; i.e. classes which can be divided by a line or hyper-plane.
  - Most functions are more complex; i.e. they are non-linear or not linearly separable.

Limitations of Simple Neural Networks
Example

More difficult problems ⟶ **More complex, multi-layer networks**





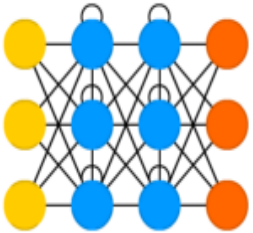
Feedforward Networks

Different Topologies of Neural Networks

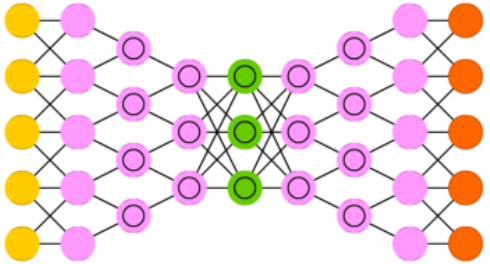
Feed Forward (FF)

Deep Feed Forward (DFF)

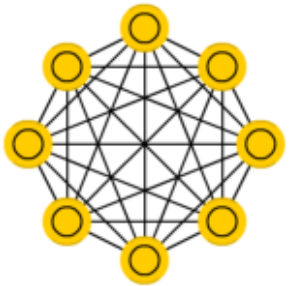
Recurrent Neural Network (RNN)



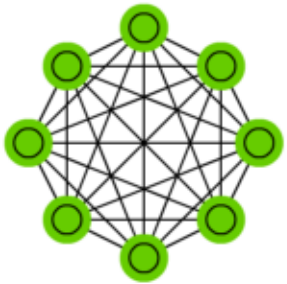
Deep Convolutional Inverse Graphics Network (DCIGN)



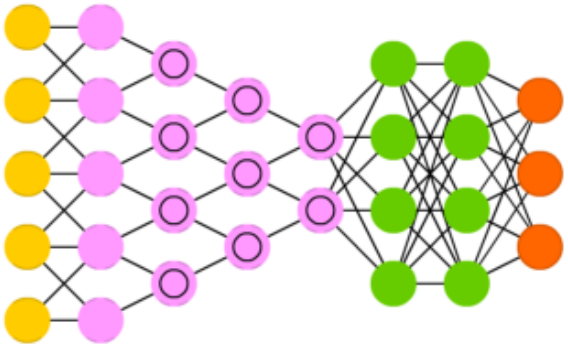
Hopfield Network (HN)



Markov Chain (MC)



Deep Convolutional Network (DCN)



...

www.asimovinstitute.org

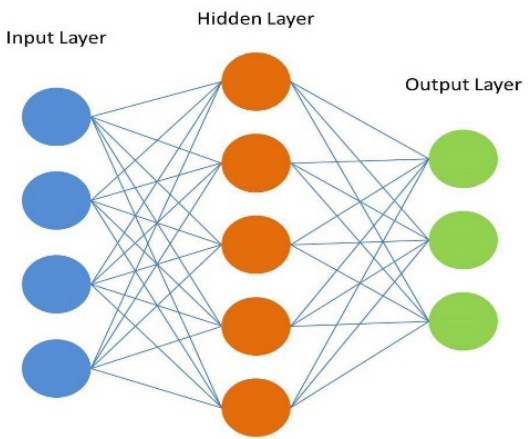
Chair of Process and Data Science

Neural Networks in This Lecture

- There are different types of neural networks
- We are going to learn about the feedforward networks
  - and later about backpropagation algorithm for this type of networks
- Feedforward networks:
  - No loops
  - input → hidden layers → output

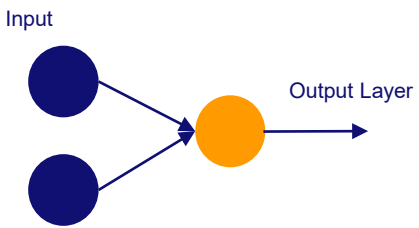
Feedforward Neural Networks

- Connection only to the next layer
- The weights of the connections (between two layers) can be changed
- Activation functions are used to calculate whether the neuron fires

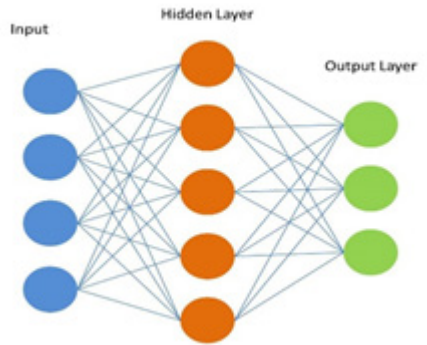


Feedforward Neural Networks
Example

- **Single-layer network:**
  - **Inputs**
  - **Output layer**



- **Two-layer network:**
  - **Inputs**
  - **Hidden layer**
  - **Output layer**



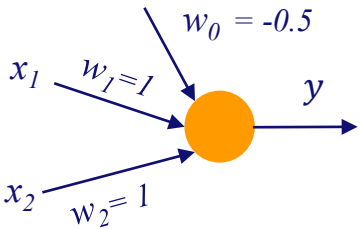
Chair of Process and Data Science

Feedforward Neural Networks
Example

Single-Layer OR

Logical OR Function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



$$w_1 x_1 + w_2 x_2 + w_0 = sum \;:\; f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$$

$$x_1 + x_2 - 0.5 = sum \;:\; f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$$

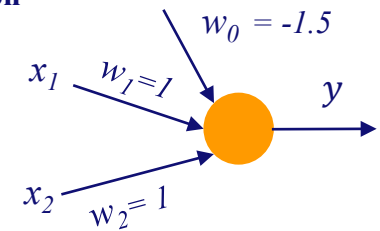
Chair of Process and Data Science

Feedforward Neural Networks
Example

Single-Layer AND

Logical AND Function

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



$$w_1 x_1 + w_2 x_2 + w_0 = sum \;:\; f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$$

$$x_1 + x_2 - 1.5 = sum \;:\; f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$$

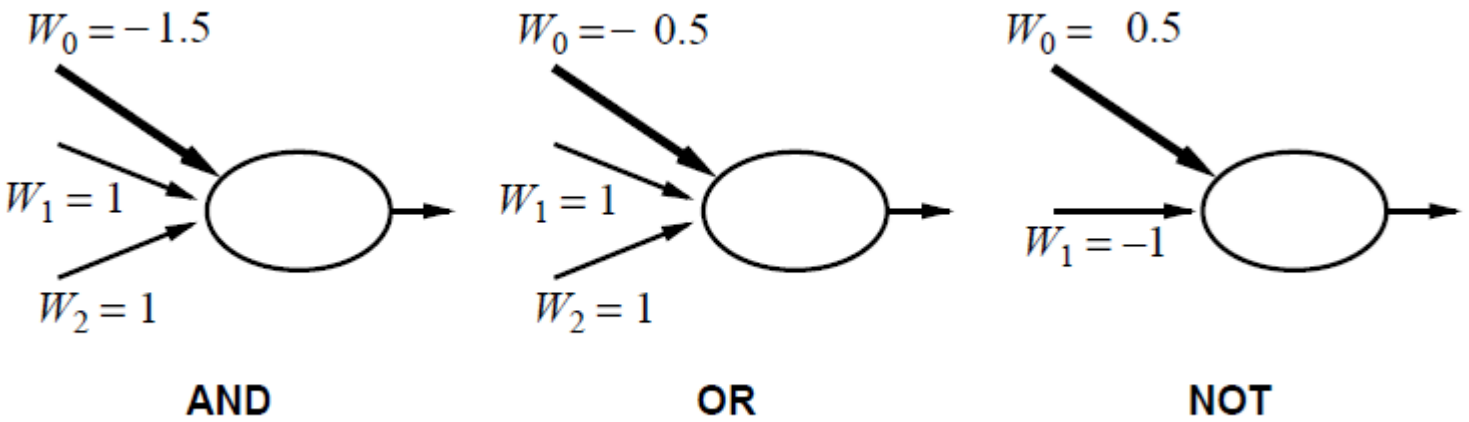
Chair of Process and Data Science

Feedforward Neural Networks
Example

$x_1 + x_2 - 1.5 = sum$: $f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$

$x_1 + x_2 - 0.5 = sum$: $f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$

$-x_1 + 0.5 = sum$: $f(sum) = \begin{cases} 0, & sum < 0 \\ 1, & sum \geq 0 \end{cases}$



Every Boolean function can be implemented,
… but not in one step.

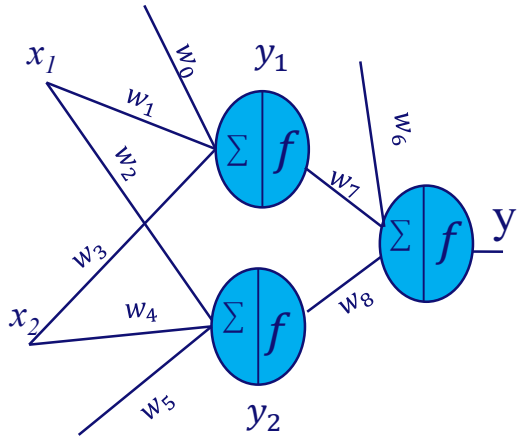
Feedforward Neural Networks

Example (Solving XOR Problem)

Logical XOR function

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Hidden layer of neurons



$$f(w_1 * x_1 + w_3 * x_2 + w_0) = y_1$$

$$f(w_2 * x_1 + w_4 * x_2 + w_5) = y_2$$

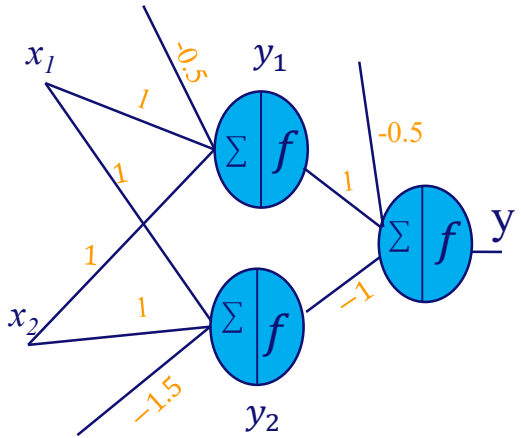
$$f(w_7 * y_1 + w_8 * y_2 + w_6) = \mathrm{y}$$

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

One more layer including two neurons is needed!
Their combined results can produce proper classifications.

Feedforward Neural Networks

Example (Solving XOR Problem)



$$f(1 * x_1 + 1 * x_2 - 0.5) = y_1 \quad \text{“or”}$$

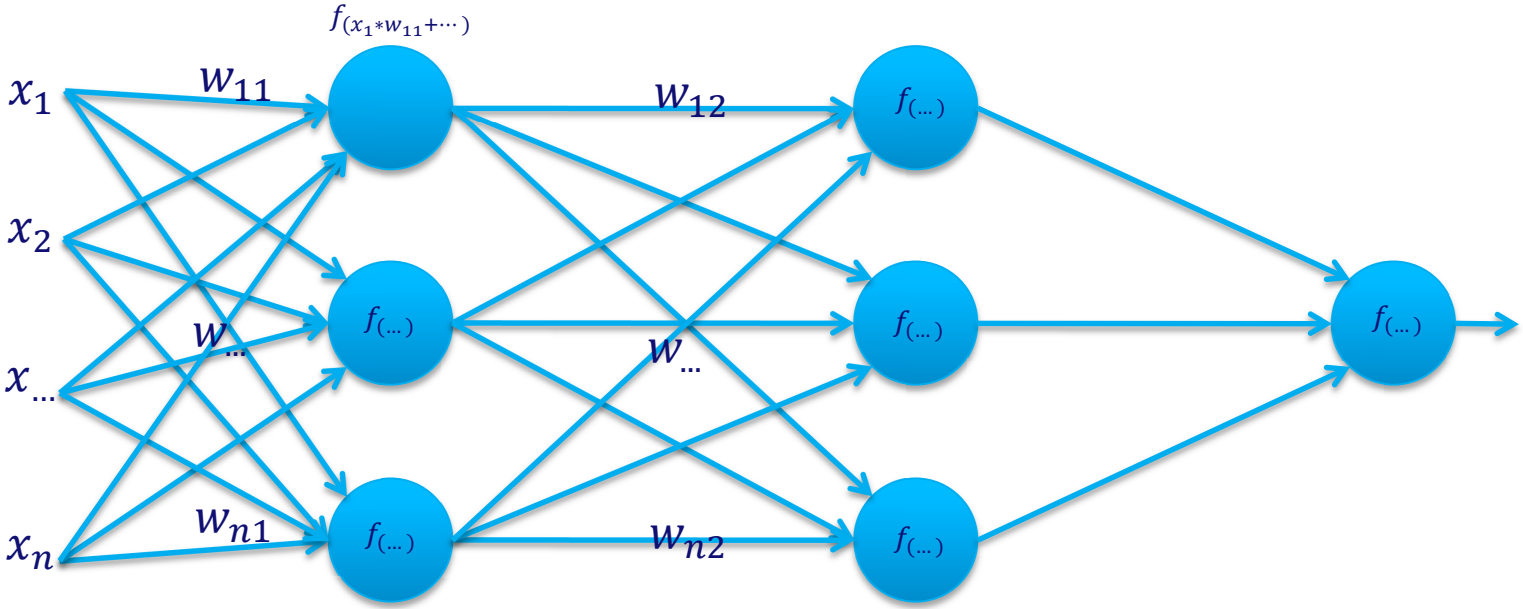
$$f(1 * x_1 + 1 * x_2 - 1.5) = y_2 \quad \text{“and”}$$

$$f(1 * y_1 - 1 * y_2 - 0.5) = y \quad \text{“and not”}$$

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

$w_0 = -0.5$
$w_1 = 1$
$w_2 = 1$
$w_3 = 1$
$w_4 = 1$
$w_5 = -1.5$
$w_6 = -1$
$w_7 = 1$
$w_8 = -0.5$

General Feedforward Network



Chair of Process and Data Science

Sample Neural Network Model

Two layer neural network

linear combination

D inputs

Shows the element is in the first layer

$$y_k(\boldsymbol{x},\boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

- f and h are activation functions:
  - they can be different functions

    for simplicity imagine there is one h function for the hidden layer
- w_{kj} shows the weight of the edge: from jth node of hidden layer to kth node of output layer
- w_{ji} shows the weight of the edge: from ith element of input to jth node of hidden layer

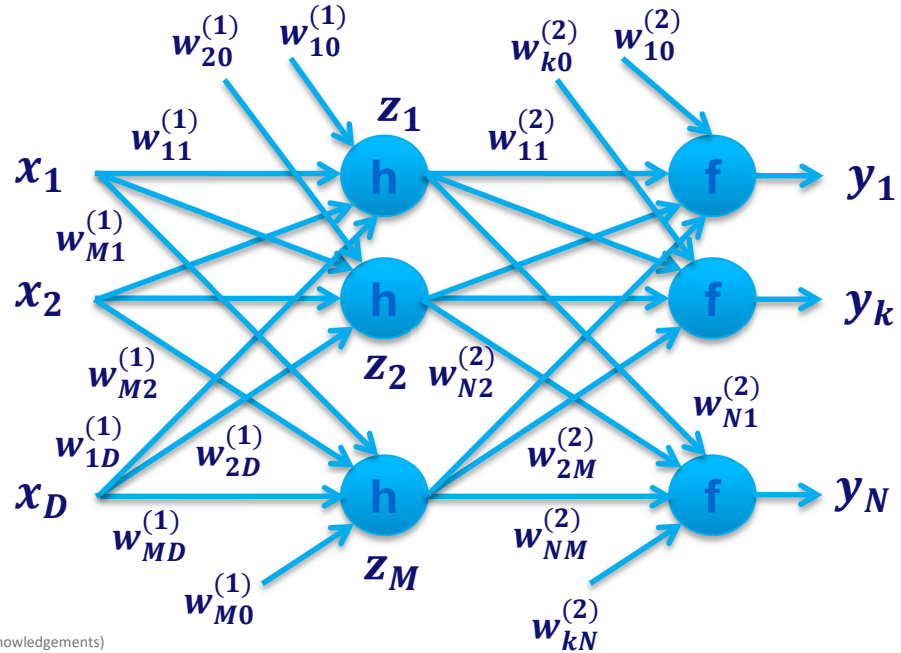
Output Layer

Chair of Process and Data Science

Sample Neural Network Model

Two layer neural network

$$y_k(\boldsymbol{x},\boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$



Chair of Process and Data Science

General Feedforward Network

Complete Two Layer Function

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\overbrace{\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}}^{z_j} \right) + w_{k0}^{(2)} \right)$$

- The hidden units with their activation functions can express non-linear functions
- The activation functions can be different at neurons

When to Use Multilayer Perceptron?

In feedforward networks

- Comprised of one or more layers of neurons
- Data is fed to the input layer
- One or more hidden layers
  - Increase level of abstraction
- Prediction based on output layer
- Suitable for
  - Classification prediction problems
    - With labeled inputs
  - Regression prediction problems
    - A real-valued quantity is predicted given a set of inputs.

When to Use Multilayer Perceptron?

In feedforward networks

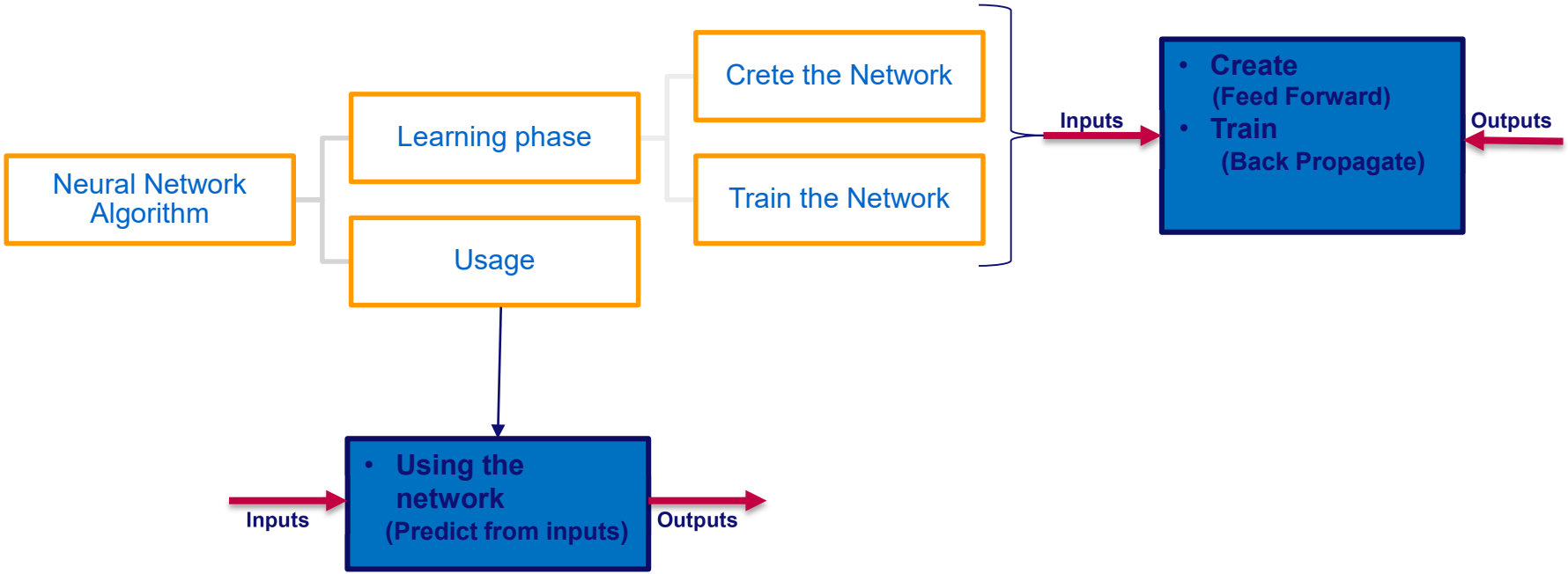
- Can be applied to different types of data
  - Text data
  - Time series data
  - Image data
    - Pixels of an image reduce to a row of data
  - Words of a document
    - Reduce to a row of data

General View of a Neural Network

- You have already learned how to create a feedforward Network
- The next step is how to train your network using error-based back-propagation approach

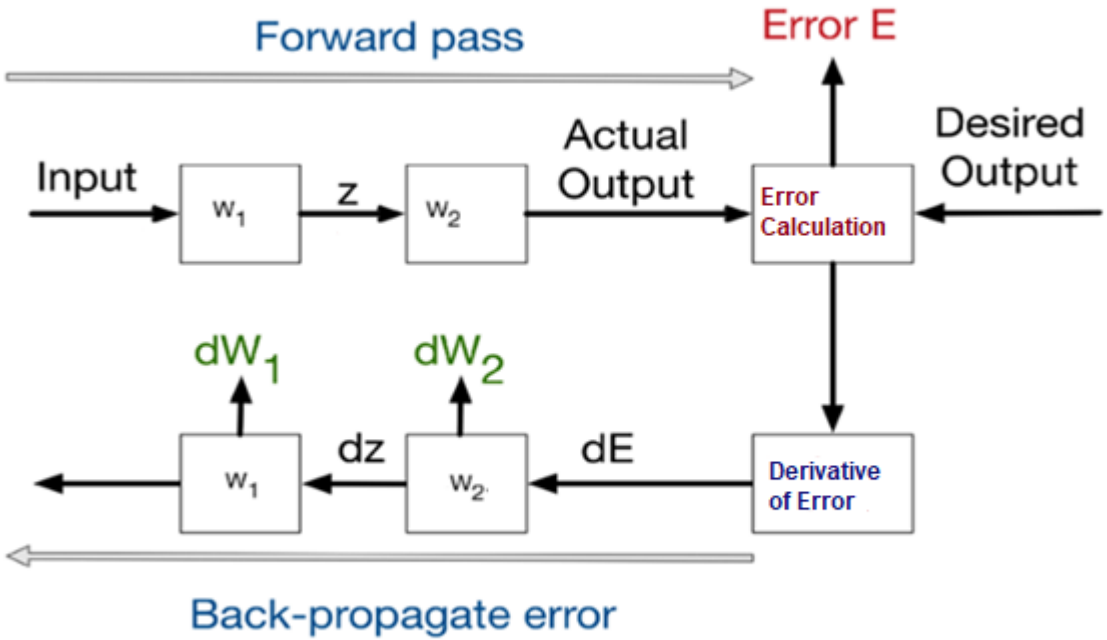
General View of a Neural Network

Training and Using



Chair of Process and Data Science

Next Lecture :
How a Neural Network Learns?
Training algorithm (Backpropagation)



Chair of Process and Data Science

Conclusion

Lecture Summary

- How to deal with non-linear problems
- A complete Neural Network elements:
  - A set of nodes (neurons)
  - A set of weight representing between each layer
  - Output and hidden layers
  - An activation function
- Network architecture
- How to build a network
- Possible usage of a simple and multilayer networks
- How to train a neural network? (Next Lecture)

#	Lecture	date	day
Lecture 1	Introduction	10/10/2018	Wednesday
Lecture 2	Crash Course in Python	11/10/2018	Thursday
Instruction 1	*Python*	*12/10/2018*	*Friday*
Lecture 3	Basic data visualisation/exploration	17/10/2018	Wednesday
Lecture 4	Decision trees	18/10/2018	Thursday
Instruction 2	*Decision trees and data visualization/exploration*	*19/10/2018*	*Friday*
Lecture 5	Regression	24/10/2018	Wednesday
Lecture 6	Support vector machines	25/10/2018	Thursday
Instruction 3	*Regression and support vector machines*	*26/10/2018*	*Friday*
Lecture 7	Neural networks (1/2)	31/10/2018	Wednesday
Instruction 4	*Neural networks and supervised learning*	*02/11/2018*	*Friday*
Lecture 8	Neural networks (2/2)	07/11/2018	Wednesday
Lecture 9	Evaluation of supervised learning problems	08/11/2018	Thursday
Instruction 5	*Neural networks and supervised learning*	*09/11/2018*	*Friday*
Lecture 10	Clustering	14/11/2018	Wednesday
Lecture 11	Frequent items sets	15/11/2018	Thursday
Lecture 12	Association rules	21/11/2018	Wednesday
Lecture 13	Sequence mining	22/11/2018	Thursday
Instruction 6	*Clustering, frequent items sets, association rules*	*23/11/2018*	*Friday*
Instruction 12	*Example exam questions*	*25/01/2018*	*Friday*
backup		30/01/2019	Wednesday
backup		31/01/2019	Thursday
extra	*Question hour*	*01/02/2019*	*Friday*