Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Week 10+11: Neural Networks

CS7CS4/CS4404 Machine Learning
v2 2018-11-20

Dr Joeran Beel
Assistant Professor in Intelligent Systems
Department of Computer Science and Statistics
Trinity College Dublin, Ireland

Dr Douglas Leith
Professor in Computer Systems
Department of Computer Science and Statistics
Trinity College Dublin, Ireland

Updates

- **Feedback for research assignment 1**
- **Guest Lecture on 27th November. Zalando. Attendance required.**

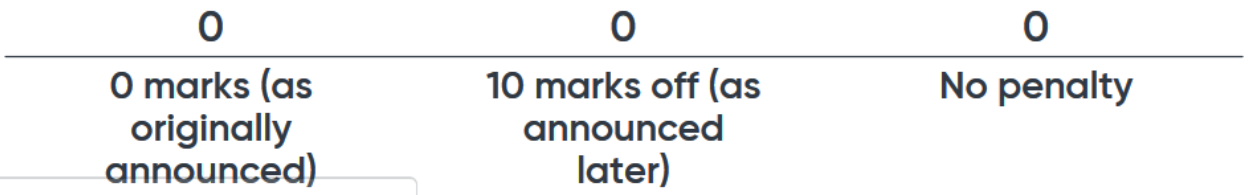
Marking / Deadline Assignment 1

- **8:00 am vs. 23:59**
  - Pro
    - Lecture Slides from first lecture
    - Previous deadline (group-preference submission)
  - Contra
    - PNG file in Onedrive
    - Doug´s deadlines
- **Bank Holiday**

Go to **www.menti.com** and use the code **50 13 9**

What would be appropriate for "late" submissions, given the "confusion" about the deadline and the general rules (0 marks)?

Mentimeter



Slide is not active **Activate**

0

Go to **www.menti.com** and use the code **50 13 9**

What would have been your preferred deadline?

Mentimeter

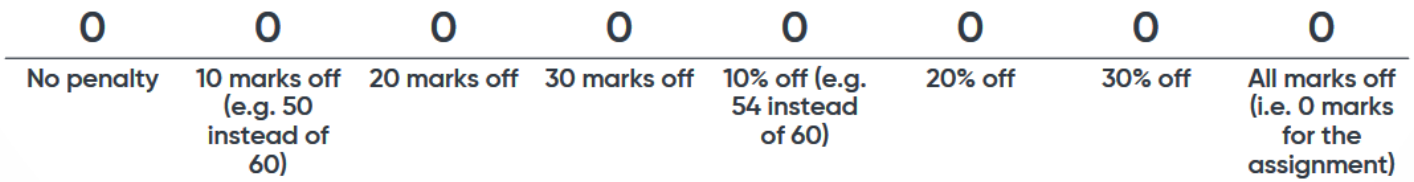
Slide is not active **Activate**

0

Go to **www.menti.com** and use the code **50 13 9**

What is an appropriate deduction for a late submission (assignment) in general?

Mentimeter



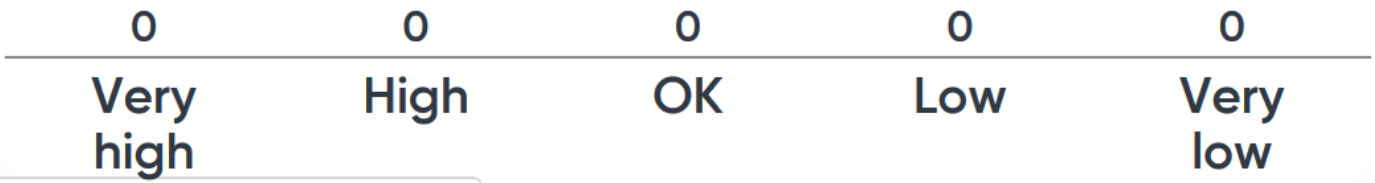
Slide is not active **Activate**

0

Go to **www.menti.com** and use the code **50 13 9**

How high is your workload is in THIS MODULE?

Mentimeter



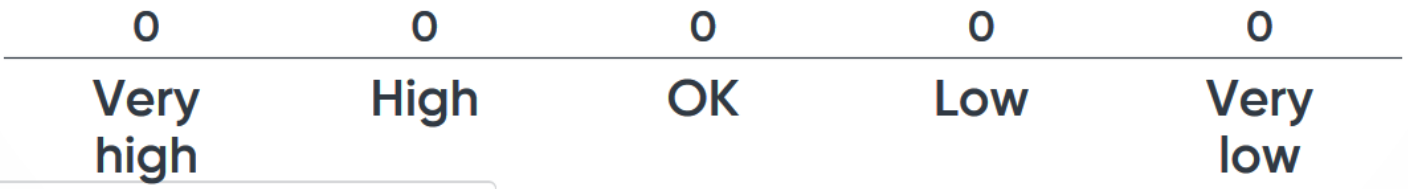
Slide is not active **Activate**

0

Go to **www.menti.com** and use the code **50 13 9**

How high is your workload overall in your COURSE (all modules in this term)?

Mentimeter



Slide is not active **Activate**

0

Artificial Neural Networks

- **Often used for classification (supervised learning)**
  - Binary Class
  - Multiclass
- **Also suitable for**
  - Regression
  - Unsupervised learning (clustering, dimensionality reduction)

History of Neural Networks (and Deep Learning)

- **Origins in 1940s**
  - Warren McCulloch (Neurophysiologist)
  - Walter Pitts (Mathematician)
- **Advances until the 1960s (including Perceptron 1958)**
- **„Dark era“ in 1970s**
- **New interest in 1980s**
- **Again, less interest in 1990s (e.g. Support Vector Machines were more promising)**
- **Since 2000s again increased interest**
  - More computing power
  - More data

Aurélien Géron, Hands on Machine Learning with scikit-learn and Tensorflow (O'Reilly Media, 2017).

Outline

1. **Biological Neural Networks**
2. **Artificial Neural Networks Overview**
3. **The McCulloch-Pitts Artificial Neuron**
4. **Perceptrons**
5. **Multi-Layer Perceptron**
6. **Practical Issues**
7. **Training/Learning Multi-Layer NN**
8. **TF-IDF**
9. **Deep Learning & Convolutional Neural Networks**

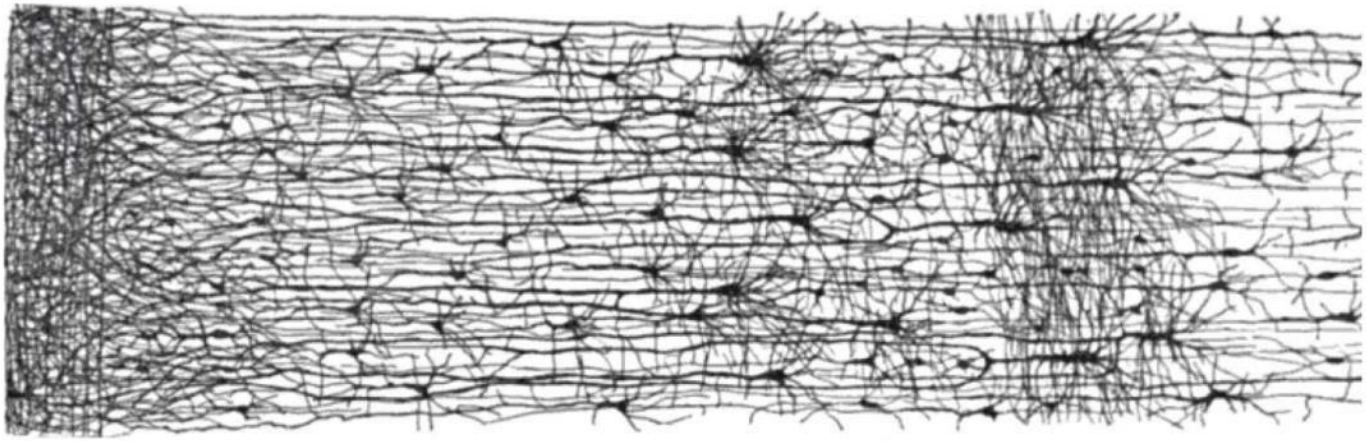
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Biological Neural Networks

The Human Neural Network

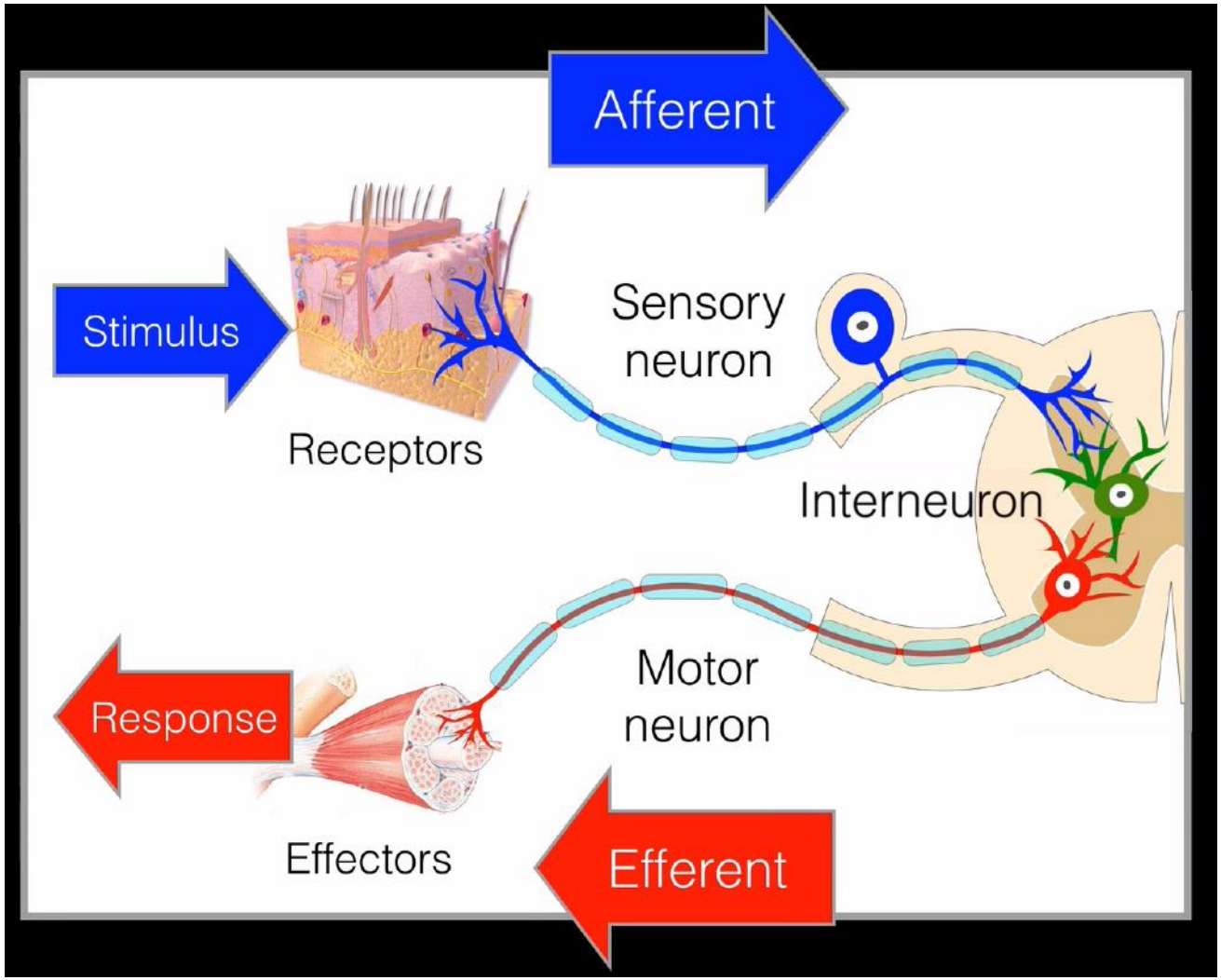
- **~ 86,000,000,000 neurons in human brain**
- **Each neuron is quite simple**
- **Each neuron is connected to ~1,000 other neurons**
- **Organized in layers**
- **The network of neurons is capable of seeing, learning, …**

- **The exact function is still widely unknown**
- **Analogy: Ants**
  - Each ant is a simple organism
  - The entire ant colony is highly organized and effective



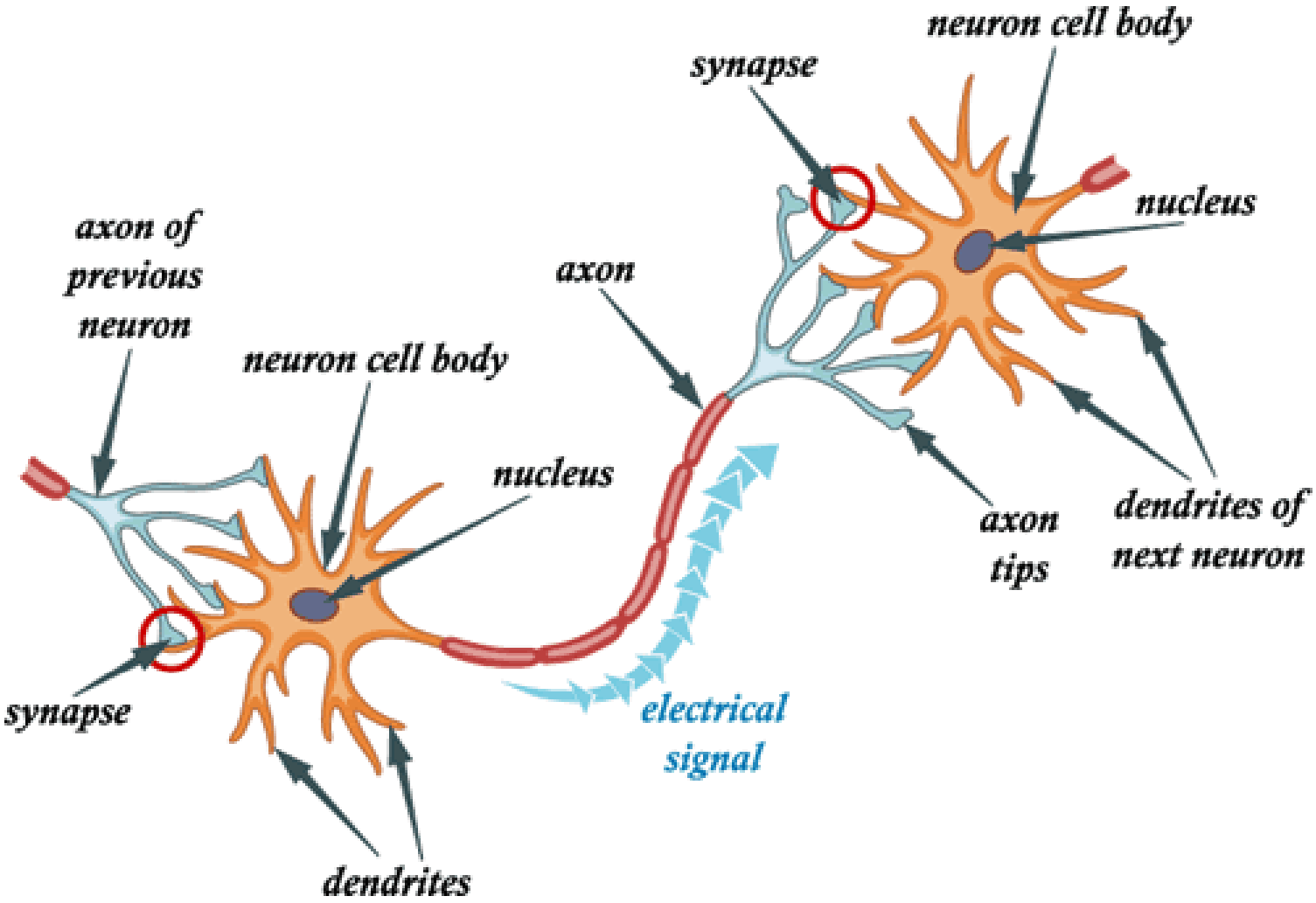
Aurélien Géron, Hands on Machine Learning with scikit-learn and Tensorflow (O'Reilly Media, 2017).

Functional Classification of Neurons



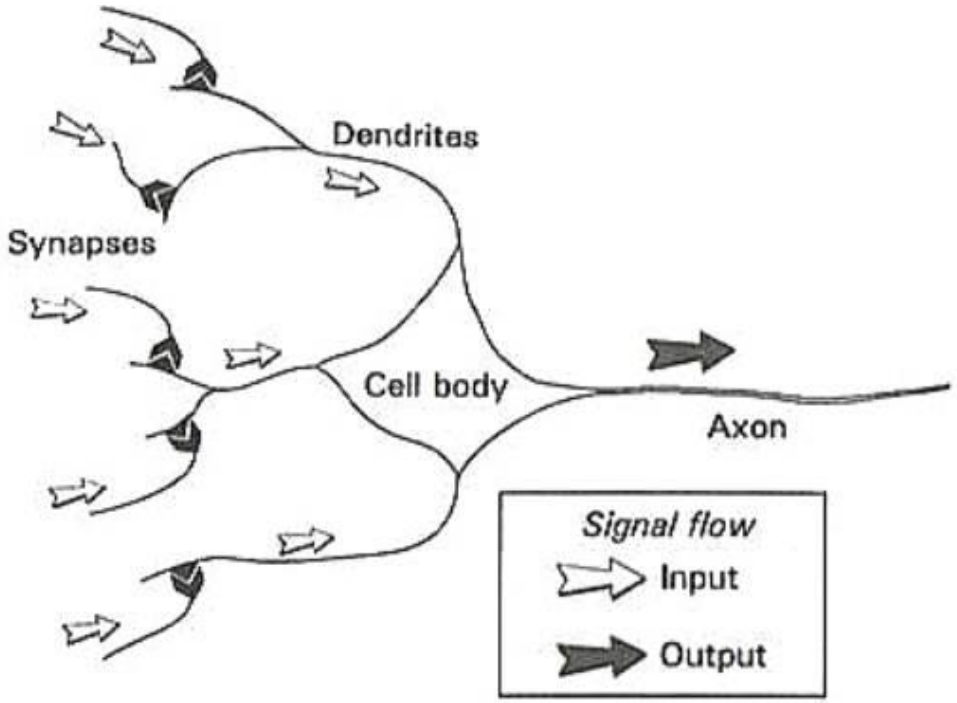
The Neuron, Paul Andersen, Bozeman Science, 2017

The Neuron / Neural Network



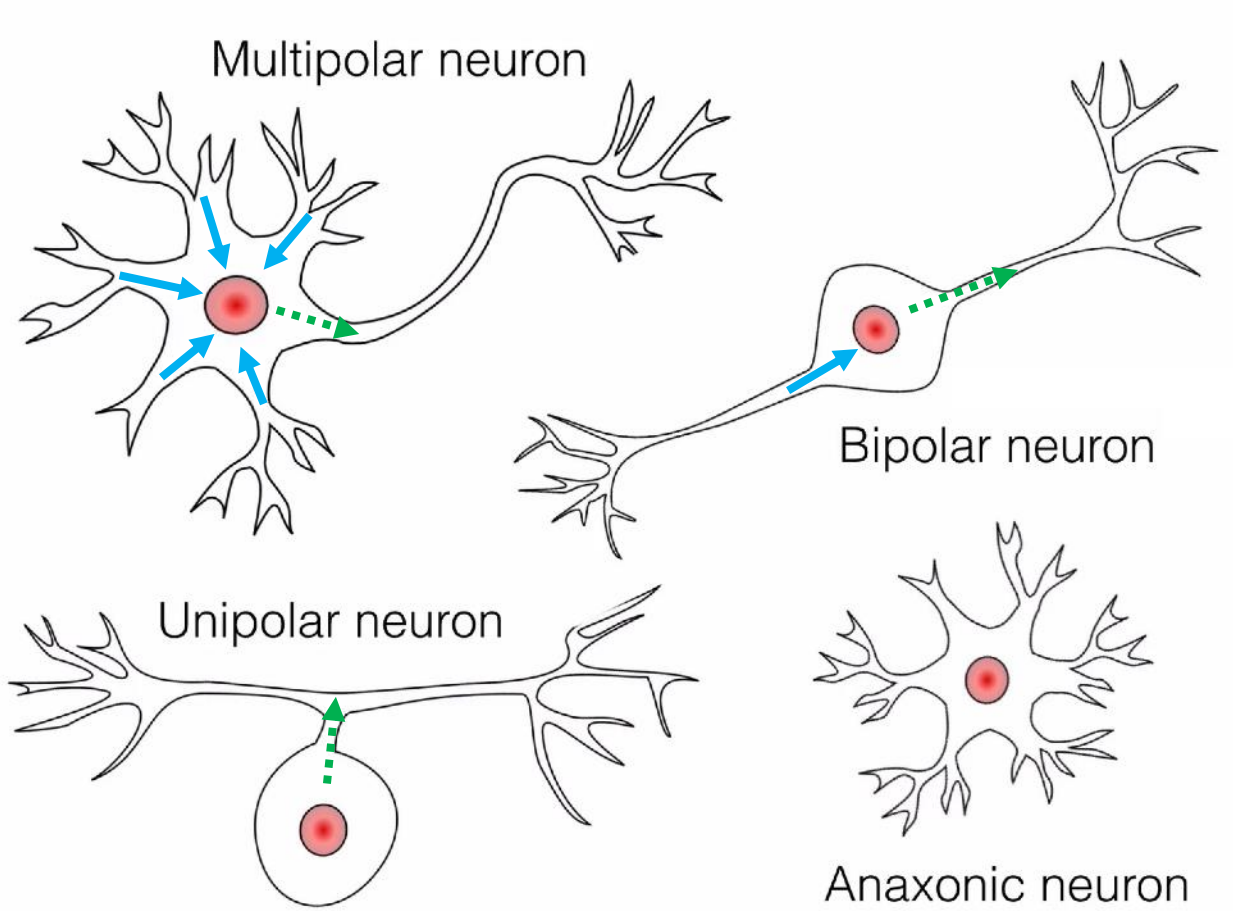
https://www.alzheimer-riese.it/images/stories/ArticlePics/Dendrites.gif

- **Output = electrical voltage**
- **Input = electrical voltage**
- **Cell body decides about output based on input**
  - Sum up inputs
  - Once a threshold is reached, output is sent
- **Output goes to other cells**
- **Connection between Dendrites and Axon = Synapsis**
- **Synapses may amplify (excitatory) or reduce the signal strength (inhibitory)**



http://aishack.in/tutorials/biological-neurons/

Structural Classification of Neurons



Cell Body (Soma)

Dendrite (excitatory or inhibitory)

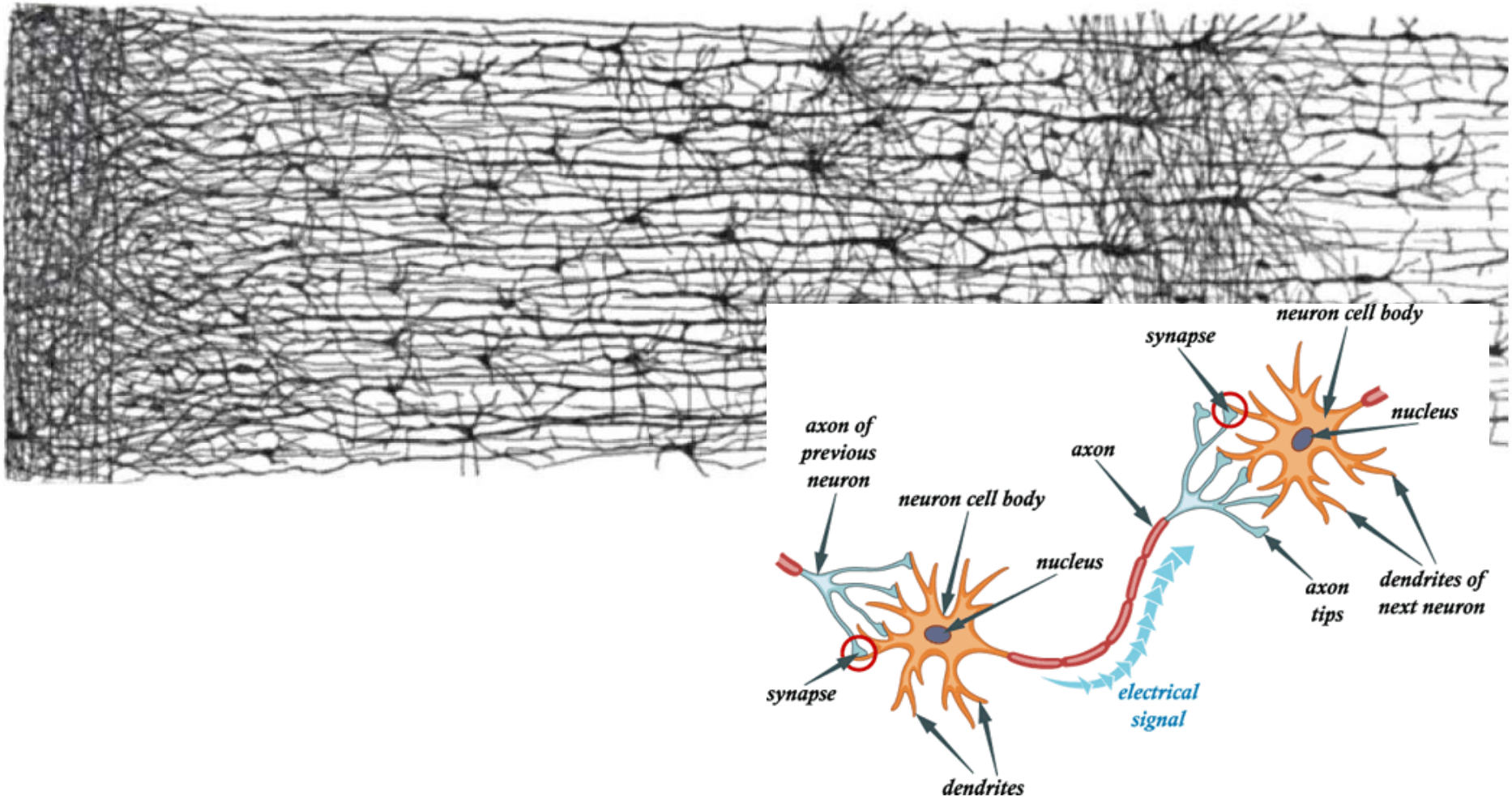
Axon

The Neuron, Paul Andersen, Bozeman Science, 2017

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

(Artificial) Neural Networks

How to re-build this in a computer?



Recap

- **Technology often adapts nature**
  - First „flight machines“ looked very much like birds
  - Most effective solutions often are not 1:1 copies
- **Analogy for learning: brain (biological/neuroscience view)**

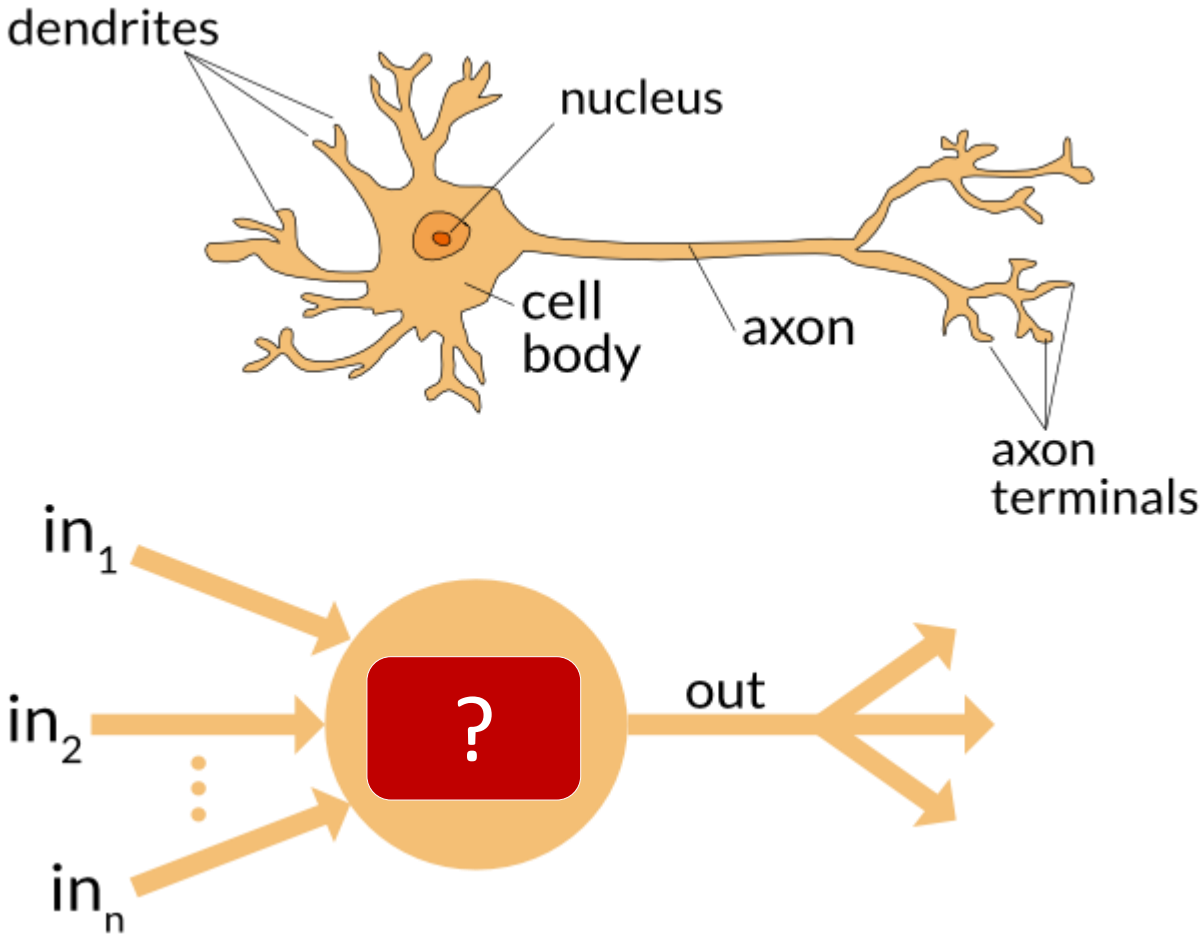
http://www.theskepticsguide.org/wp-content/uploads/2015/09/EarlyFlight.jpg

http://kookbang.dema.mil.kr/newspaper/news/20120326/38723.jpg

Tribe	Origins	Master Algorithm
Symbolists	Logic, philosophy	Inverse deduction
Connectionists	Neuroscience	Backpropagation
Evolutionaries	Evolutionary biology	Genetic programming
Bayesians	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

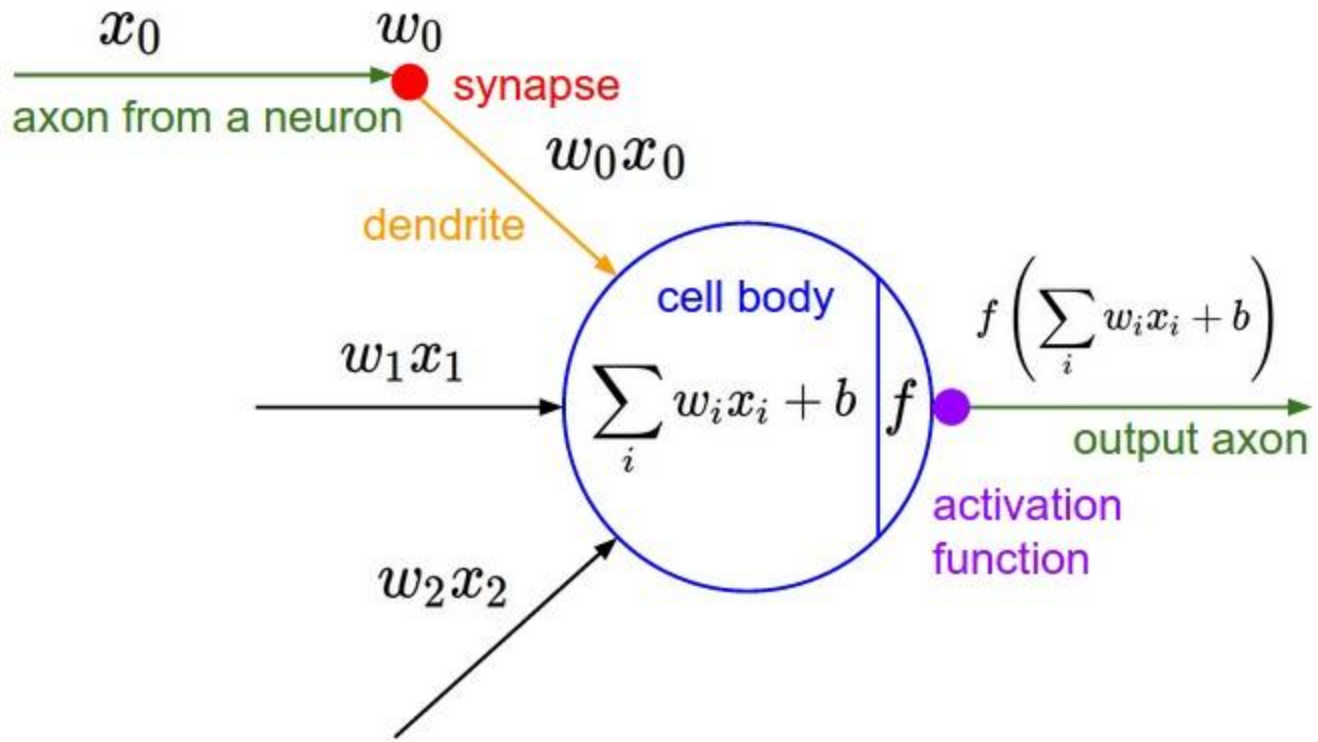
Pedros Domingos, The Five Tribes of Machine Learning: And What you Can Take From Each

Artificial and Biological Neuron Compared (1)



https://appliedgo.net/perceptron/

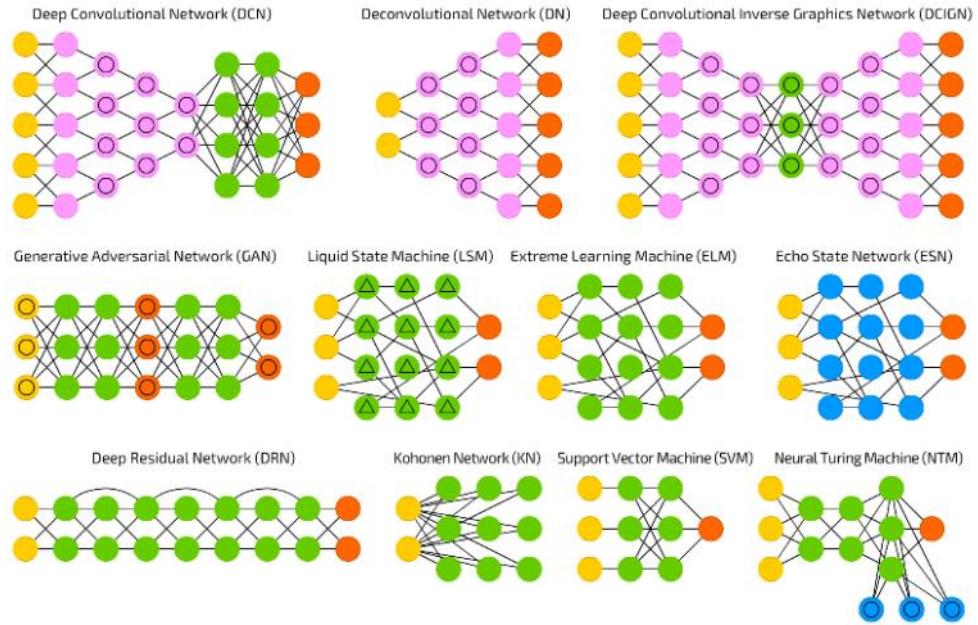
Artificial and Biological Neuron Compared (2)

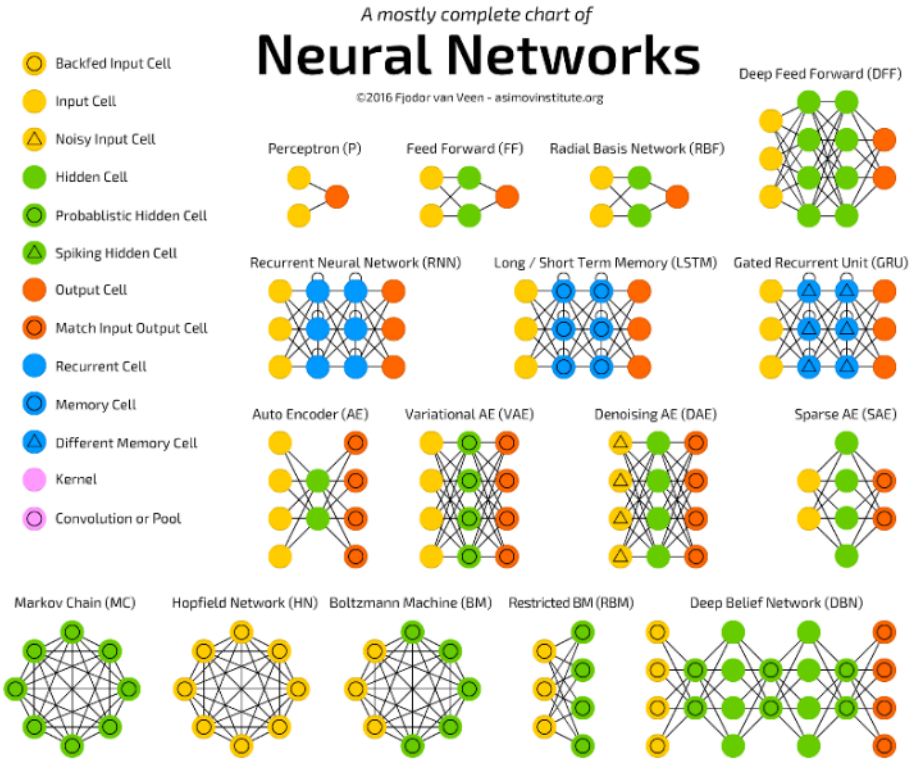


http://cs231n.github.io/neural-networks-1/

Network Strutures

- **Different Neuron Types**
- **Different Network Types**





http://www.asimovinstitute.org/wp-content/uploads/2016/09/neuralnetworks.png

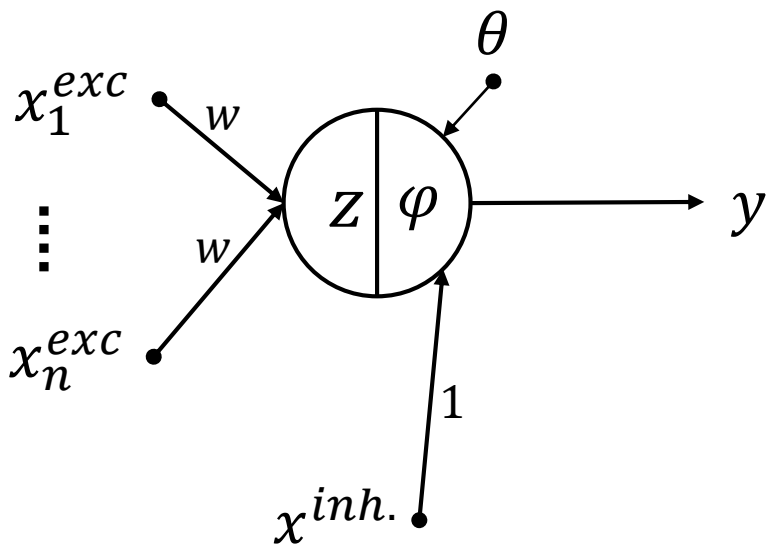
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

The McCulloch-Pitts Artificial Neuron ~~Network~~ Model, 1943

The first model of an Artificial Neuron

Warren McCulloch and Walter Pitts, 1943

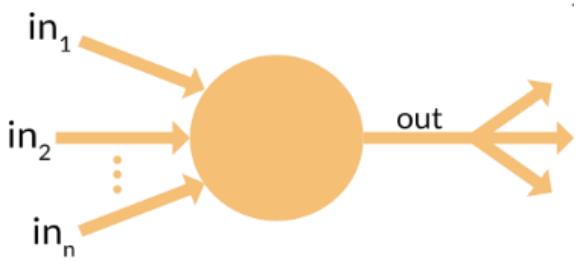
- x^{exc}: **n binary inputs from excitatory „dendrites“**
- x^{inh} **: 1 binary input from an inhibitory „dendrite“ (a „veto“ dendrite).**
- w: **weight for all excitatory inputs. Typically** $w = [0,1]$



- z: **Combination Function / Net Input =** $\sum_{i=1}^{n} w x_i^{exc}$
- φ: **Step Activation Function**

$$h(z) = \begin{cases} 1, & z \geq \theta \; AND x^{inh} = 0 \\ 0, & otherwise \end{cases}$$

- θ: **Threshold**
- y: **Binary output**



“AND“ „OR“ and „XOR“ as Classification Problem

Practical Example

$'AND'$

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

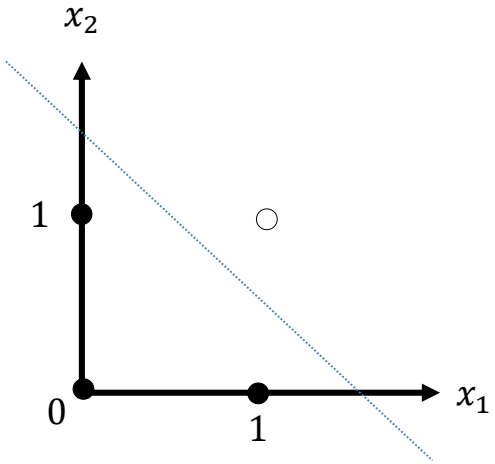
$'OR'$

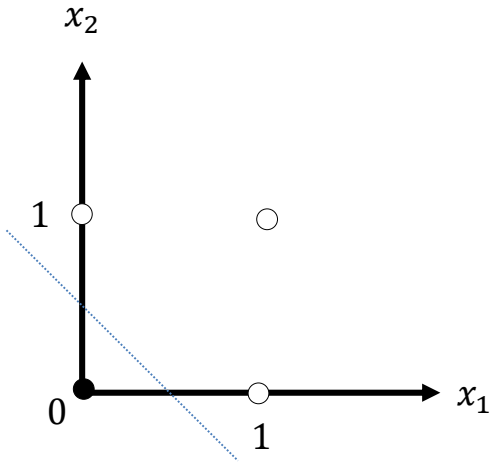
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

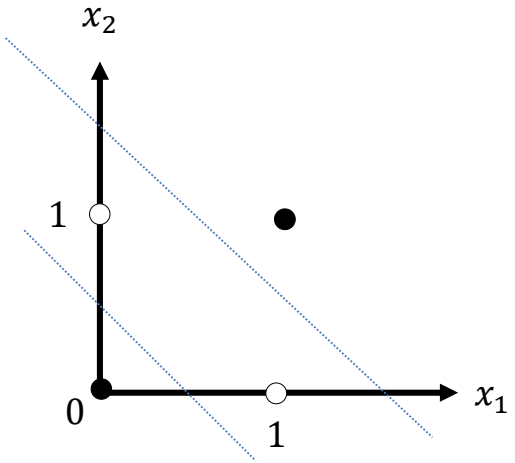
$'XOR'$

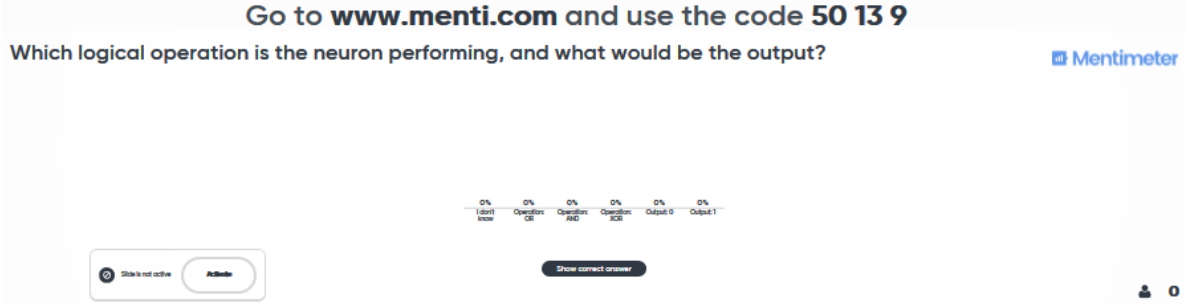
x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

○ True
● False
▪▪▪▪ Decision Boundary









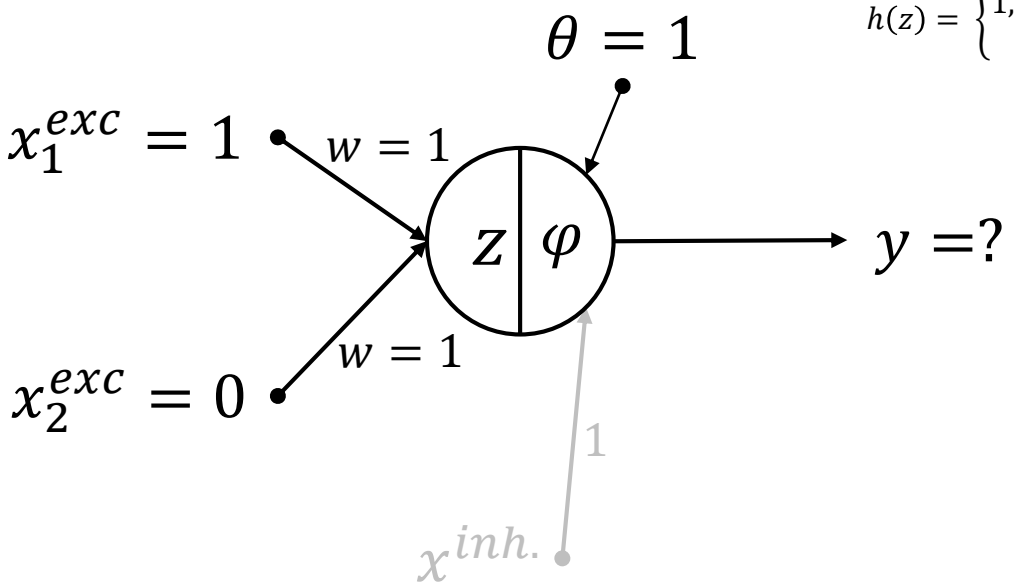
Example 1

Which logical operation is the neuron performing, and what would be the output, given the particular inputs? (we ignore the inhibitory input for now; i.e. it is 0)

$$x_i^{exc} = \{0,1\}$$

$$z = \sum_{i=1}^{n} w x_i^{exc}$$

$$h(z) = \begin{cases} 1, & z \geq \theta \; AND x^{inh} = 0 \\ 0, & otherwise \end{cases}$$



$'OR'$		
x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

Go to **www.menti.com** and use the code **50 13 9**

Which logical operation is the neuron performing, and what would be the output?

Mentimeter

0%	0%	0%	0%	0%	0%
I don't know	Operation: OR	Operation: AND	Operation: XOR	Output: 0	Output: 1

Slide is not active **Activate**

0

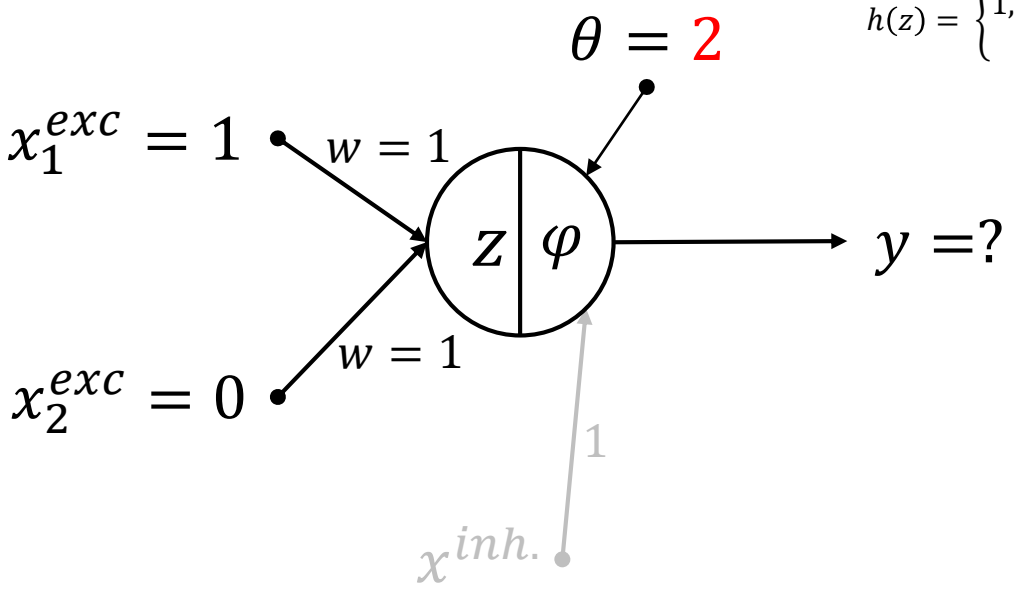
Example 2

Which logical operation is the neuron performing, and what would be the output, given the particular inputs? (we ignore the inhibitory input for now; i.e. it is 0)

$$x_i^{exc} = \{0,1\}$$

$$z = \sum_{i=1}^{n} w x_i^{exc}$$

$$h(z) = \begin{cases} 1, & z \geq \theta \; AND x^{inh} = 0 \\ 0, & otherwise \end{cases}$$



		'AND'
x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Go to **www.menti.com** and use the code **50 13 9**

Which logical operation is the neuron performing, and what would be the output?

Mentimeter

0%	0%	0%	0%	0%	0%
I don't know	Operation: OR	Operation: AND	Operation: XOR	Output: 0	Output: 1

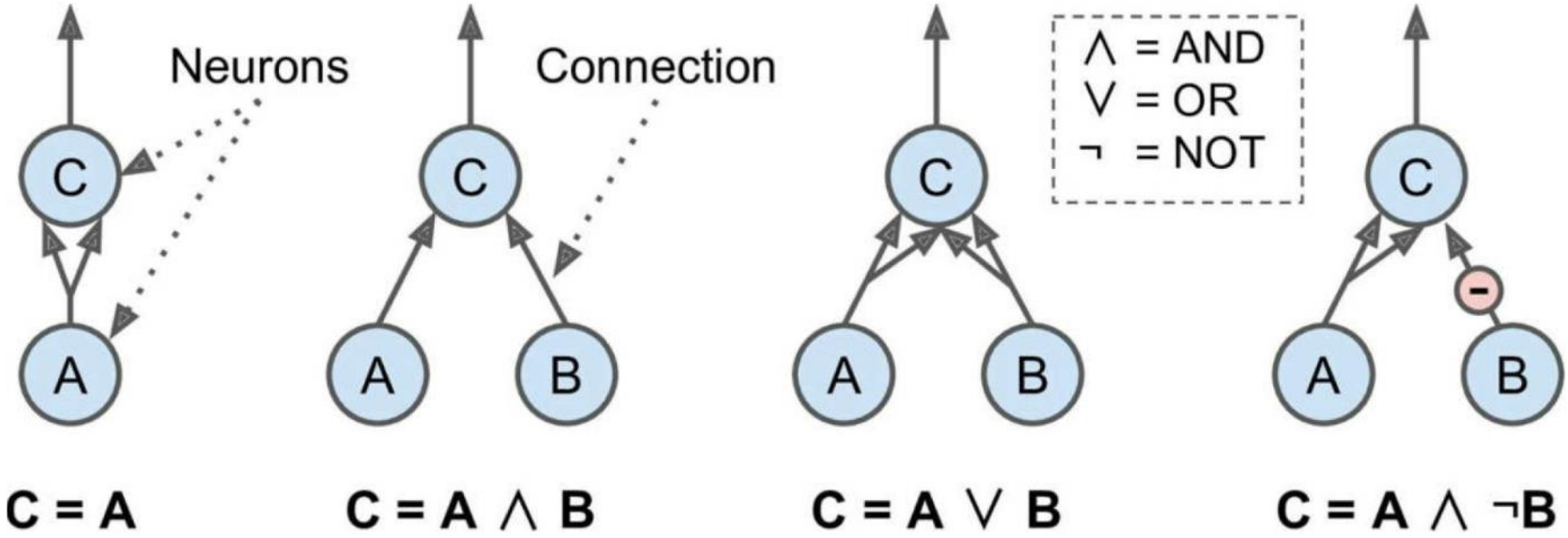
Show image

Slide is not active **Activate**

0

Variations

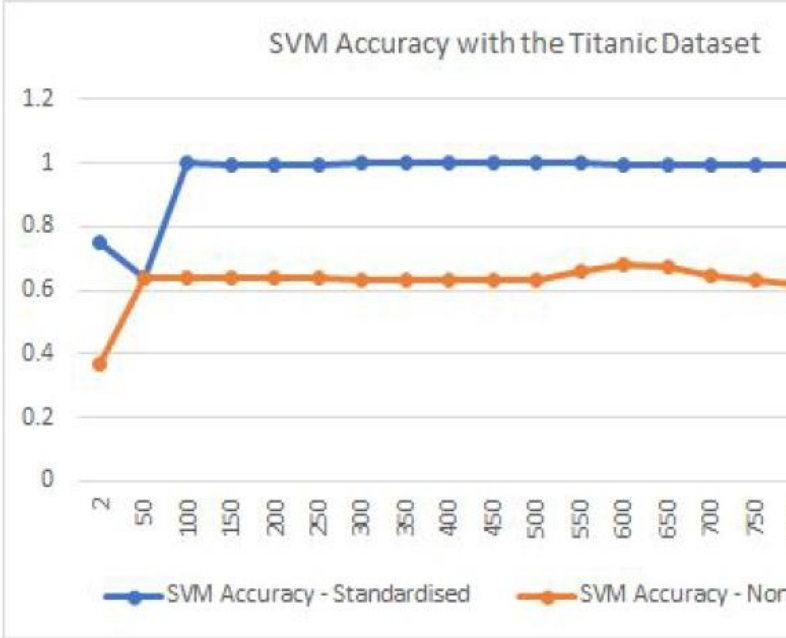
Neuron is activated if two inputs are active $(\theta = 2)$



Aurélien Géron, Hands on Machine Learning with scikit-learn and Tensorflow (O'Reilly Media, 2017).

Update

- **Accuracy of (near) 100%**
- **Potential reasons**
  - Just a very good classifier / simple data
  - Data Leakage
    - Test data included in training
    - Target variable included in training
    - „Target Twin“ included in training (that would not be available in the real world); can be identified through correlation analysis (or thinking)
    - Future instances included (especially for time-series prediction)
- **(Note: The term “Data Leakage” is also used in the context of adversarial ML but then with a different meaning)**



Serving Customer Insights in Zalando (Upcoming Presentation in our Machine Learning Lecture)

Published by **Joeran Beel** on 19th November 2018

After announcing a guest presentation from Zalando in our e-Business lecture, we are delighted to announce another talk by Zalando, this time in our machine learning lecture.

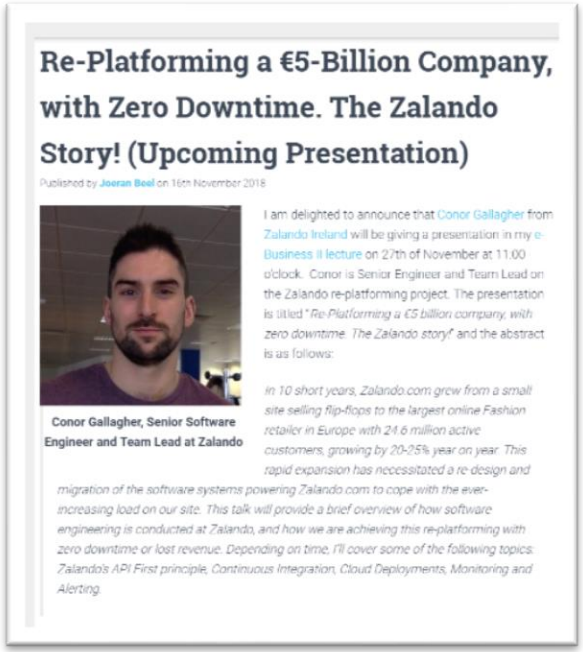
Antoaneta Marinova, Data Scientist and Machine-Learning Engineer at Zalando

Antoaneta Marinova from Zalando, will give a presentation on 27th November at 15:00 o'clock. Antoaneta is a Data Engineer in the Customer Fashion Profile team. She works at Zalando for two years, mainly on attribute recommendations and customer segmentation. Previously, she was working on ad optimisation for Adcash and as a software developer in Axway. She has a masters degree in Artificial Intelligence and a bachelor in Computer Science from Sofia University.

Antoaneta´s talk is titled "**Serving Customer Insights in Zalando**". The abstract is as follows.

The presentation will focus on Zalando's Customer Fashion Profile. The team provides insights and empower personalisation by identifying fresh, relevant and impactful attributes for Zalando customers. I will introduce how we define customer problems, provide online

https://www.scss.tcd.ie/joeran.beel/blog/2018/11/19/serving-customer-insights-in-zalando-upcoming-presentation-in-our-machine-learning-lecture/



Re-Platforming a €5-Billion Company, with Zero Downtime. The Zalando Story! (Upcoming Presentation)

Published by Joeran Beel on 16th November 2018

Conor Gallagher, Senior Software Engineer and Team Lead at Zalando

I am delighted to announce that Conor Gallagher from Zalando Ireland will be giving a presentation in my e-Business II lecture on 27th of November at 11:00 o'clock. Conor is Senior Engineer and Team Lead on the Zalando re-platforming project. The presentation is titled "*Re-Platforming a €5 billion company, with zero downtime. The Zalando story!*" and the abstract is as follows:

In 10 short years, Zalando.com grew from a small site selling flip-flops to the largest online Fashion retailer in Europe with 24.6 million active customers, growing by 20-25% year on year. This rapid expansion has necessitated a re-design and migration of the software systems powering Zalando.com to cope with the ever-increasing load on our site. This talk will provide a brief overview of how software engineering is conducted at Zalando, and how we are achieving this re-platforming with zero downtime or lost revenue. Depending on time, I'll cover some of the following topics: Zalando's API First principle, Continuous Integration, Cloud Deployments, Monitoring and Alerting.

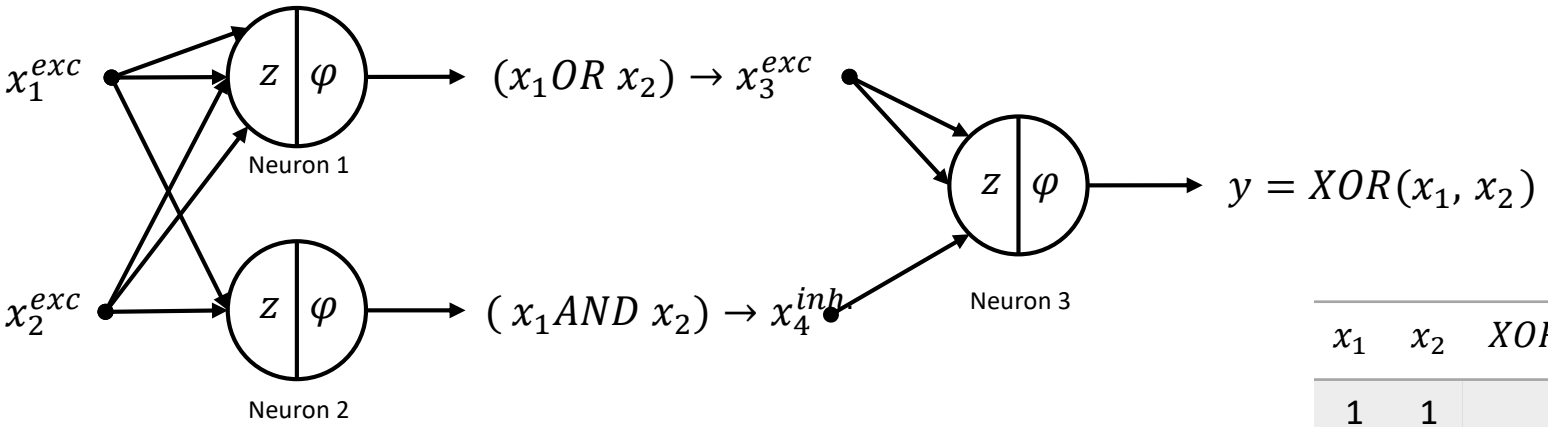
https://www.scss.tcd.ie/joeran.beel/blog/2018/11/16/re-platforming-a-e5-billion-company-with-zero-downtime-the-zalando-story/

XOR

$$XOR(x_1, x_2) = (x_1 OR\ x_2)\ AND\ NOT\ (x_1 AND\ x_2)$$
$$w = 1$$
$$\theta = 2$$

$$z = \sum_{i=1}^{n} w x_i^{exc}$$

$$h(z) = \begin{cases} 1, & z \geq \theta\ AND x^{inh} = 0 \\ 0, & otherwise \end{cases}$$



x_1	x_2	$XOR(x_1, x_2)$
1	1	0
1	0	1
0	1	1
0	0	0

Where is the learning?

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Perceptrons

Frank Rosenblatt 1958

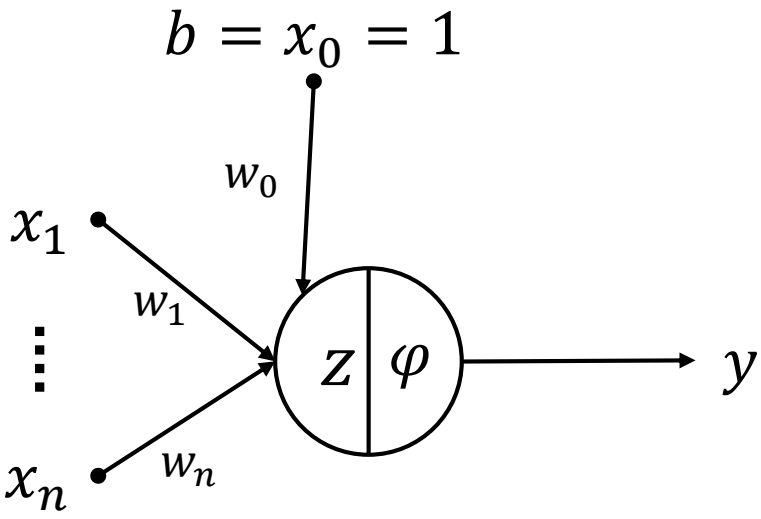
Neuron / Linear Threshold Unit (LTU)

- ***n* variable inputs $x_{1\ldots n}$**
- ***x* are features, not instances**
- **One constant input $x_0 = 1$ („bias feature / bias neuron")**
- ***n+1* weights $w_{0\ldots n}$**
- **One output y (a network of LTUs may have multiple outputs, one for each LTU)**
- **Input and output are often, but not necessarily, binary**
- **No special inhibitory input**
- **Training Goal: Learn $w_{0\ldots n}$**

$$z = \sum_{i=0}^{n} w_i x_i$$

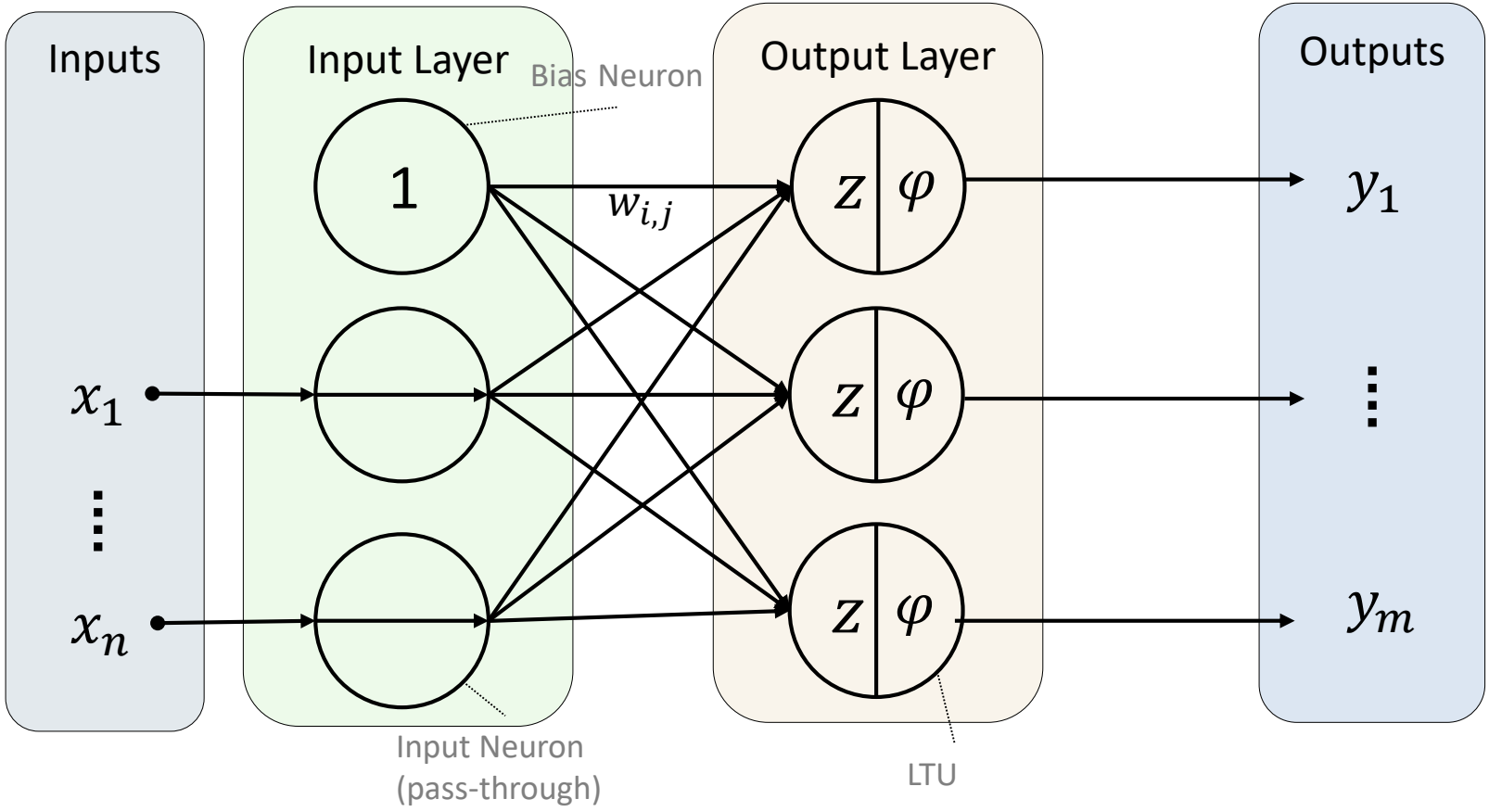
$$\varphi_{heaviside} = h(z) = \begin{cases} 0\ (or\ -1)\ if\ z < 0 \\ 1\ if z \geq 0 \end{cases}$$

$$\varphi_{sign} = sgn(z) = \begin{cases} -1\ if z < 0 \\ 0\ if\ z = 0 \\ 1\ if\ z > 0 \end{cases}$$



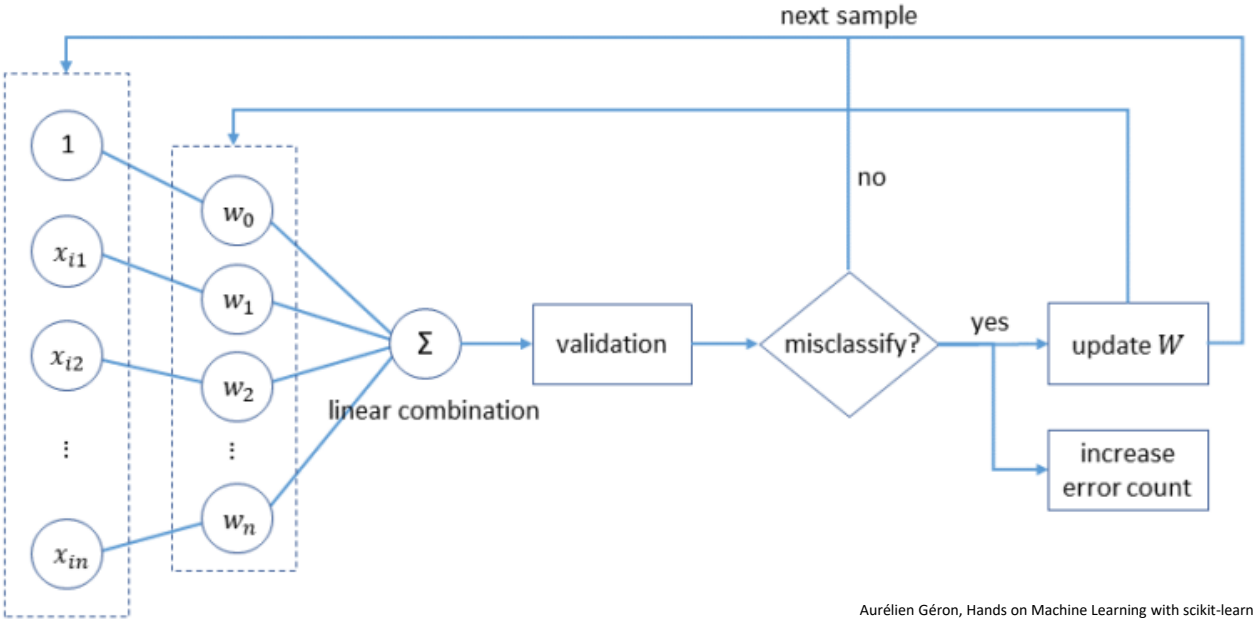
Single Layer Perceptron (i.e. a single layer of LTUs)

- **One Input Layer**
- **One Output Layer**
- **Input neurons just pass through the inputs**
- **Each neuron in layer l_k is connected to each neuron in l_{k+1}**
- **Bias neuron always outputs 1**
- **Image shows 2 inputs, 3 outputs – Multioutput classifier**



Perceptron Training (1)

- **Based on Hebb´s rule / Hebbian learning (1949)**
- **Connections between biological neurons that trigger each other frequently, become stronger over time**
- **“Cells that fire together, wire together.” (Siegrid Löwel)**
- **Weight between two neurons is increased when they have the same output**
- **No reinforcement for wrong output**
- **The perceptron looks at each training instance at a time**



Aurélien Géron, Hands on Machine Learning with scikit-learn and Tensorflow (O’Reilly Media, 2017).
https://www.codeproject.com/Articles/1211753/Machine-Learning-Basics-and-Perceptron-Learning-Al

Perceptron Training (2)

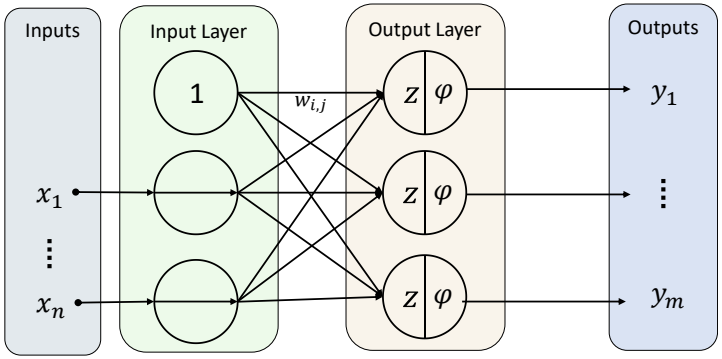
$$(y_j - \hat{y}_j) = Error$$

$$\varphi = \begin{cases} -1 \; if \; z < 0 \\ 1 \; if z \geq 0 \end{cases}$$

- **Initialize weights $w_{i,j}$ with small random numbers [-1 , 1]**
- **For each training instance**
  - Calculate output with current weights
  - Update weights (if prediction was wrong) :

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

- **Continue until:**
  - Convergence (for linearly sep. classes)
  - Error threshold
  - Fixed number of iterations



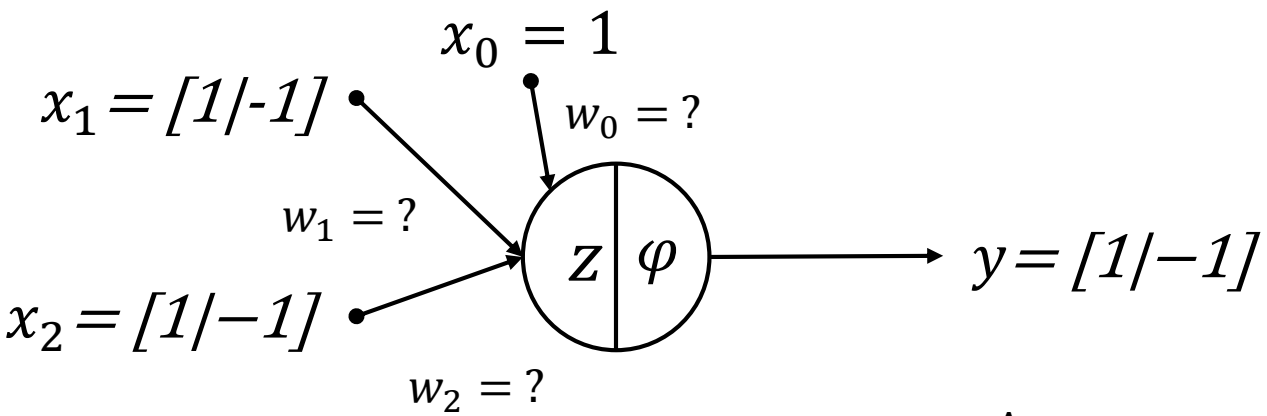
y	$\hat{y}$	$\Delta w_{i,j}$
1	1	$\eta(1 - 1)\, x_i = 0$
-1	-1	$\eta(-1 - -1)\, x_i = 0$
1	-1	$\eta(1 - -1)\, x_i = \eta(2)x_i$
-1	1	$\eta(-1 - 1)\, x_i = \eta(-2)x_i$

x_i	i^{th} input value
$w_{i,j}$	weight between i^{th} input neuron and j^{th} output neuron
$\hat{y}_j$	output of j^{th} output neuron
y_j	target output of j^{th} output neuron
η	learning rate (constant between 0 and 1); "eta"

Example ('AND')

x_1	x_2	y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

Epoch	x_0	x_1	x_2	y	w_0	w_1	w_2	∑	ŷ	Error	Converged?	η
1	1	-1	-1	-1	0.1	0.1	0.2	-0.2	-1	0		0.2
	1	-1	1	-1	0.1	0.1	0.2	0.2	1	-2		
	1	1	-1	-1	-0.3	0.5	-0.2	0.4	1	-2		
	1	1	1	1	-0.7	0.1	0.2	-0.4	-1	2	Not Converged	
2	1	-1	-1	-1	-0.3	0.5	0.6	-1.4	-1	0		
	1	-1	1	-1	-0.3	0.5	0.6	-0.2	-1	0		
	1	1	-1	-1	-0.3	0.5	0.6	-0.4	-1	0		
	1	1	1	1	-0.3	0.5	0.6	0.8	1	0	Converged	



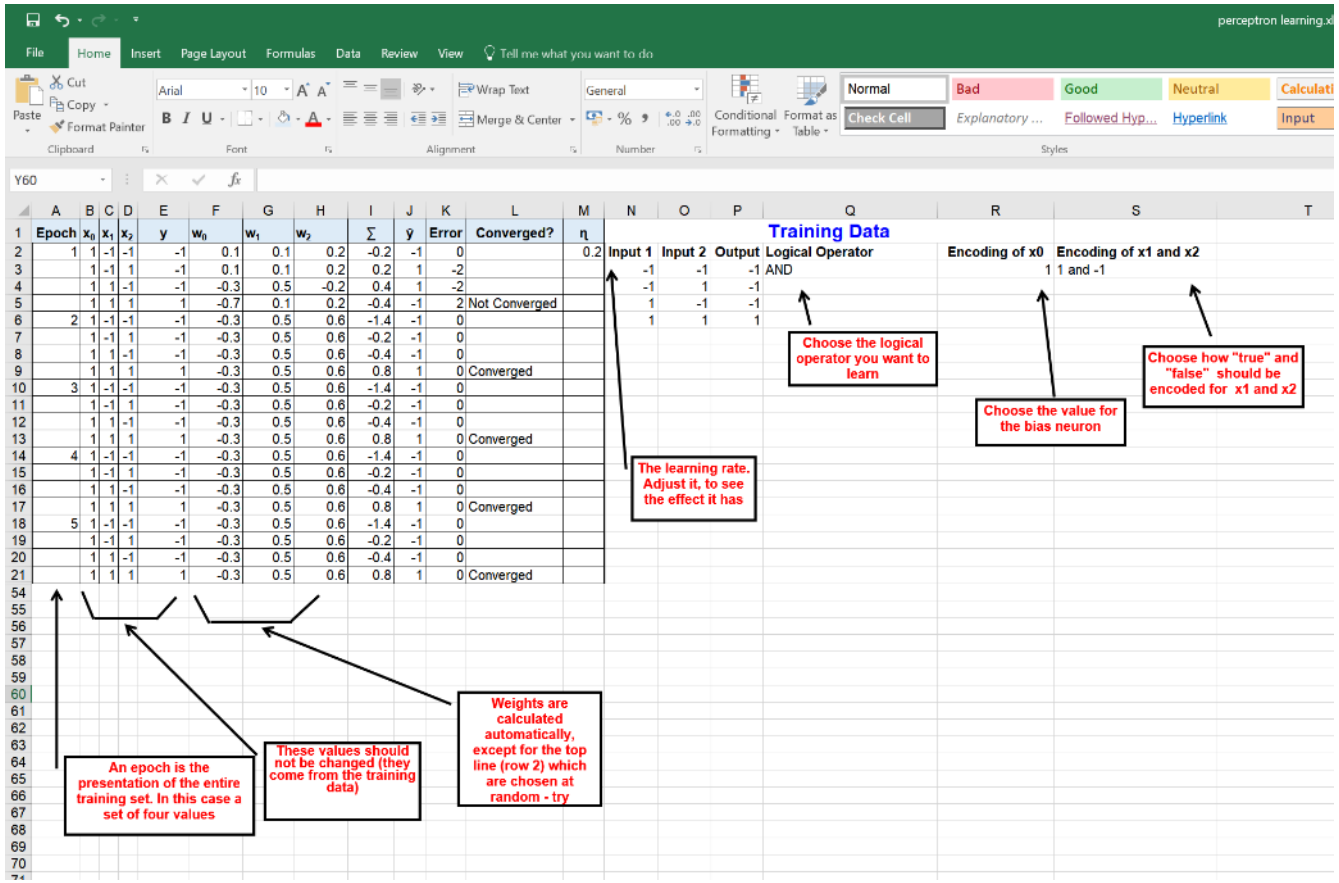
$$z = \sum_{i=0}^{n} w_i x_i$$

$$\varphi = \begin{cases} -1 \; if \; z < 0 \\ 1 \; if z \geq 0 \end{cases}$$

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

Excel Tool

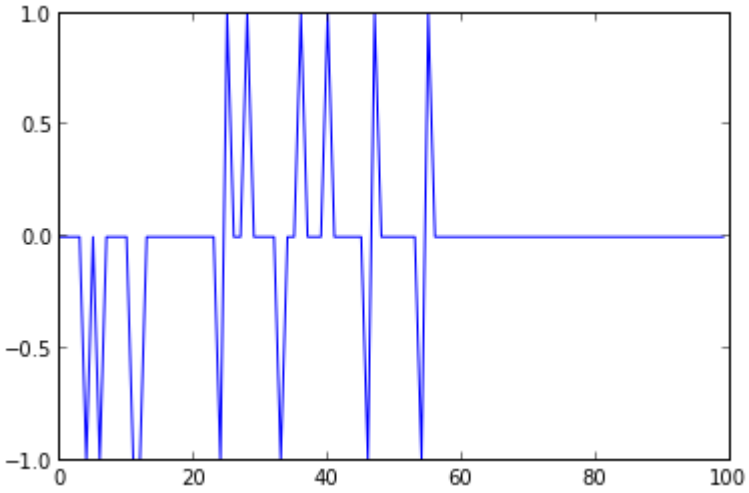
- **Excel File on Blackboard (lecture slides, „perceptron learning.xlsx“)**



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Epoch	x_0	x_1	x_2	y	w_0	w_1	w_2	$\sum$	$\hat{y}$	Error	Converged?	η
2	1	1	-1	-1	-1	0.1	0.1	0.2	-0.2	-1	0		0.2
3		1	-1	1	-1	0.1	0.1	0.2	0.2	1	-2		
4		1	1	-1	-1	-0.3	0.5	-0.2	0.4	1	-2		
5		1	1	1	1	-0.7	0.1	0.2	-0.4	-1	2	Not Converged	
6	2	1	-1	-1	-1	-0.3	0.5	0.6	-1.4	-1	0		
7		1	-1	1	-1	-0.3	0.5	0.6	-0.2	-1	0		
8		1	1	-1	-1	-0.3	0.5	0.6	-0.4	-1	0		
9		1	1	1	1	-0.3	0.5	0.6	0.8	1	0	Converged	
10	3	1	-1	-1	-1	-0.3	0.5	0.6	-1.4	-1	0		
11		1	-1	1	-1	-0.3	0.5	0.6	-0.2	-1	0		
12		1	1	-1	-1	-0.3	0.5	0.6	-0.4	-1	0		
13		1	1	1	1	-0.3	0.5	0.6	0.8	1	0	Converged	
14	4	1	-1	-1	-1	-0.3	0.5	0.6	-1.4	-1	0		
15		1	-1	1	-1	-0.3	0.5	0.6	-0.2	-1	0		
16		1	1	-1	-1	-0.3	0.5	0.6	-0.4	-1	0		
17		1	1	1	1	-0.3	0.5	0.6	0.8	1	0	Converged	
18	5	1	-1	-1	-1	-0.3	0.5	0.6	-1.4	-1	0		
19		1	-1	1	-1	-0.3	0.5	0.6	-0.2	-1	0		
20		1	1	-1	-1	-0.3	0.5	0.6	-0.4	-1	0		
21		1	1	1	1	-0.3	0.5	0.6	0.8	1	0	Converged	

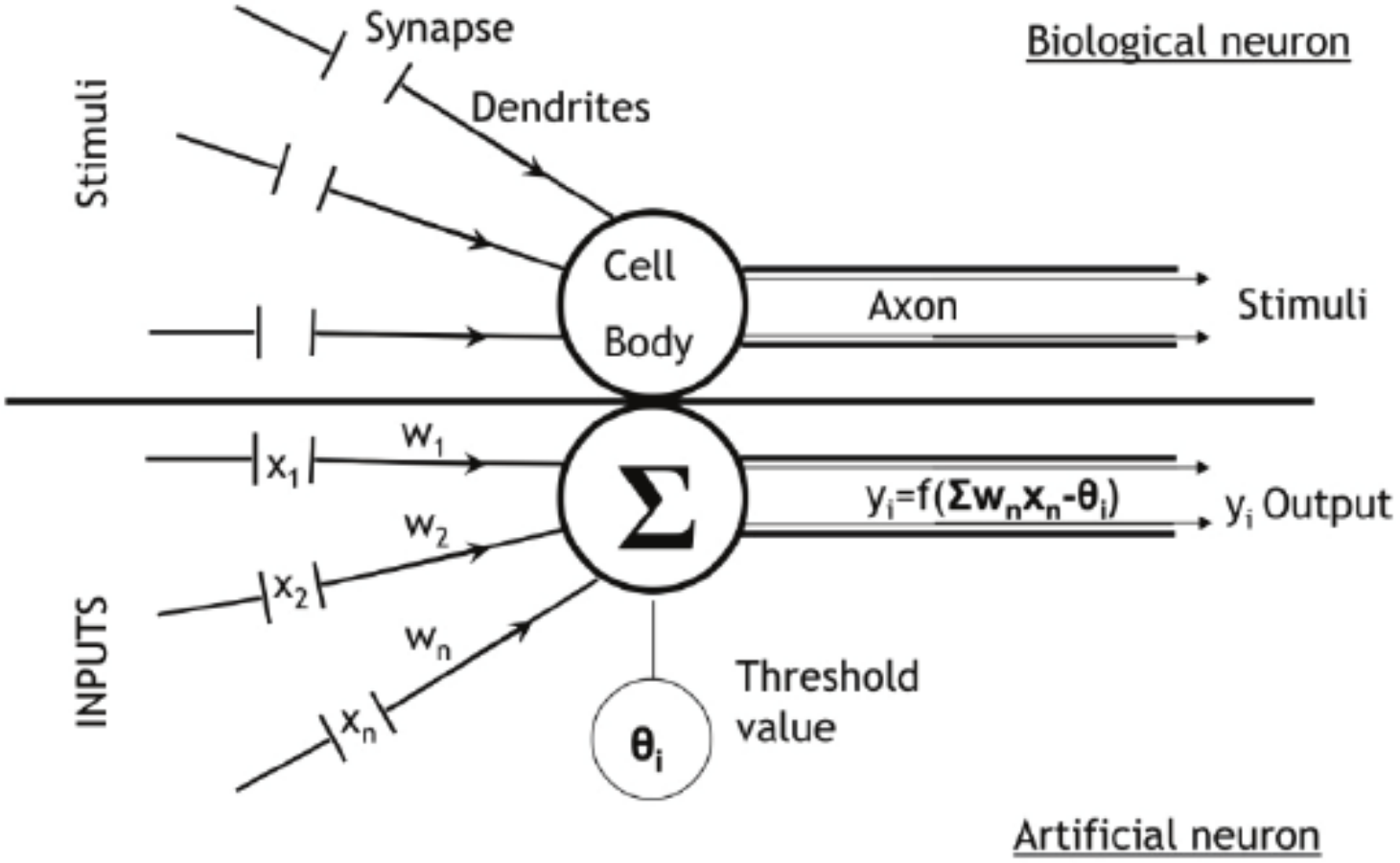
A Perceptron in Python

```
from random import choice
from numpy import array, dot, random

unit_step = lambda x: 0 if x < 0 else 1

training_data = [
                (array([0,0,1]), 0),
                (array([0,1,1]), 1),
                (array([1,0,1]), 1),
                (array([1,1,1]), 1), ]

w = random.rand(3)
errors = []
eta = 0.2
n = 100

for i in xrange(n):
                x, expected = choice(training_data)
                result = dot(w, x)
                error = expected - unit_step(result)
                errors.append(error)
                w += eta * error * x


for x, _ in training_data:
                result = dot(x, w)
                print("{}: {} -> {}".format(x[:2], result, unit_step(result)))
```



https://blog.dbrgn.ch/2013/3/26/perceptrons-in-python/

Perceptron and Biological Neuron Compared



Pedro Pablo Gallego et al. 2011. Artificial Neural Networks Technology to Model and Predict Plant Biology Process. *Artificial Neural Networks - Methodological Advances and Biomedical Applications.*

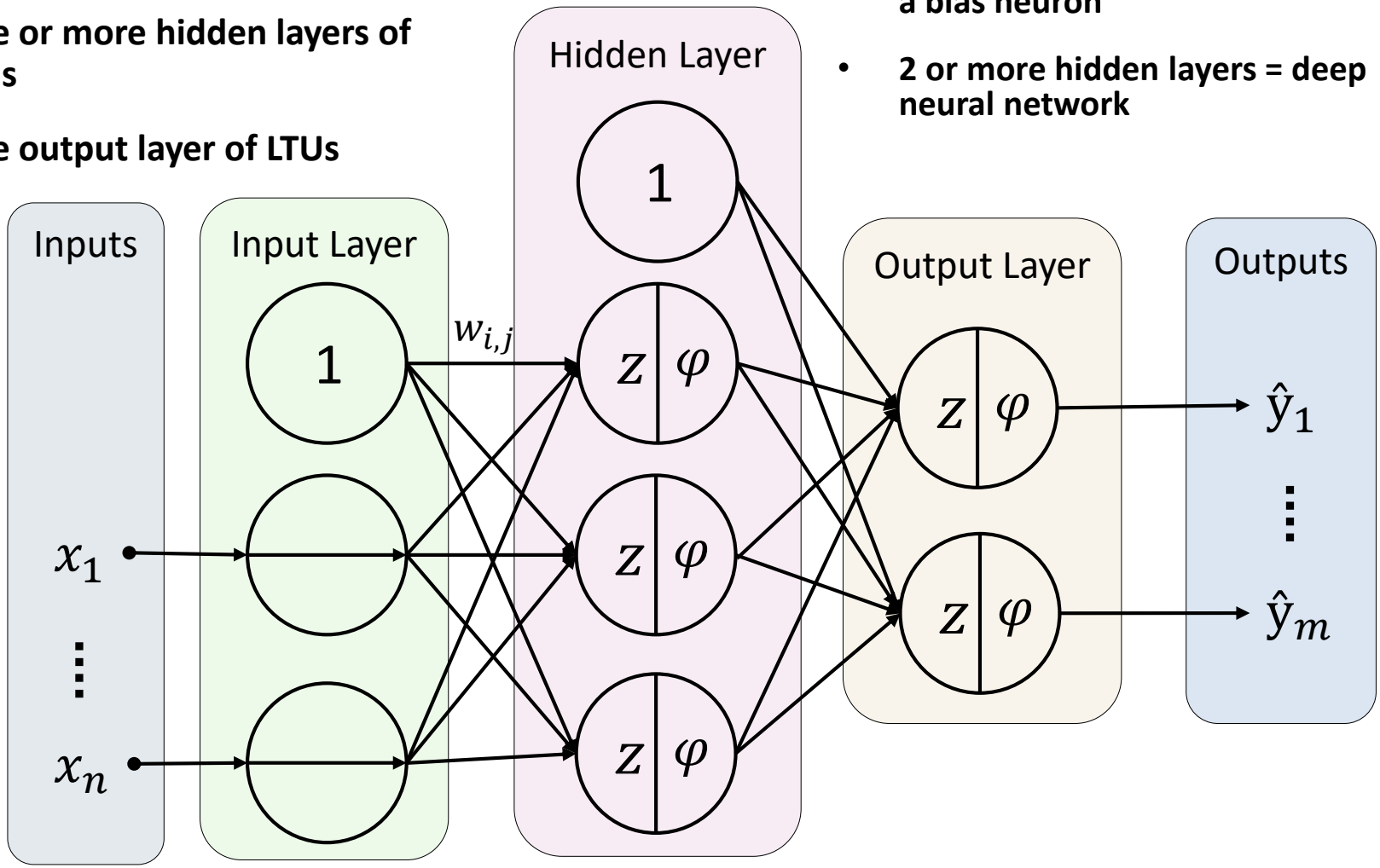
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Multi-Layer Perceptrons (MLP), Overview

Overview

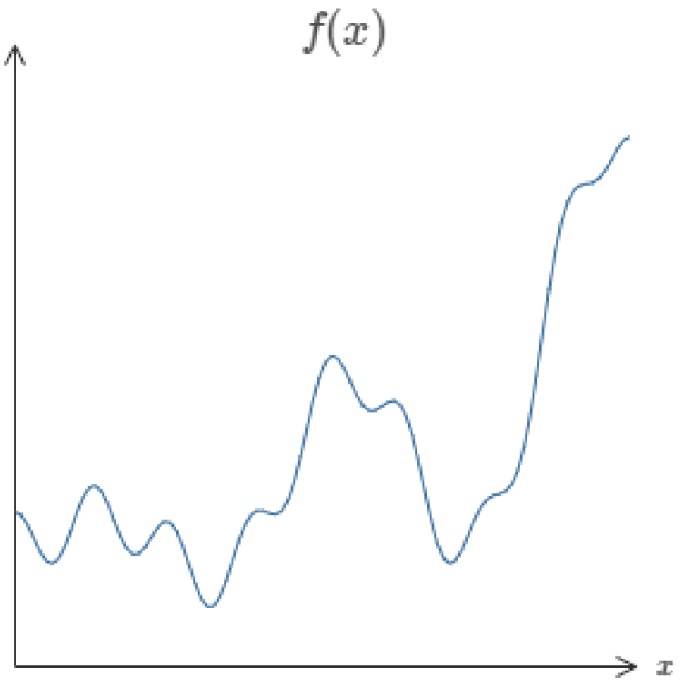
- **One passthrough input layer**
- **One or more hidden layers of LTUs**
- **One output layer of LTUs**

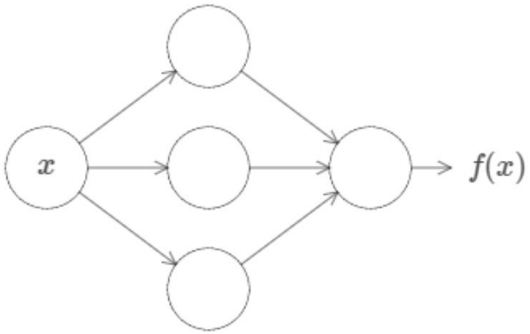
- **Input and hidden layers include a bias neuron**
- **2 or more hidden layers = deep neural network**

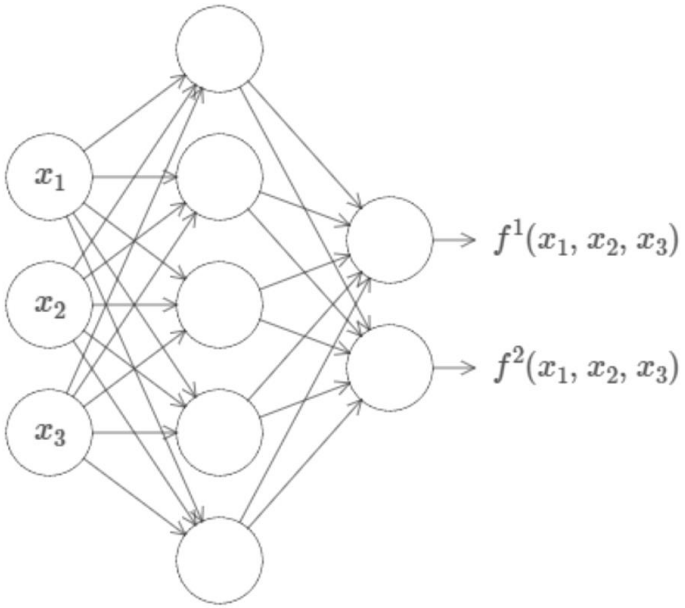


Do whatever you want

- **With one hidden layer, any mathematical function can be modelled**

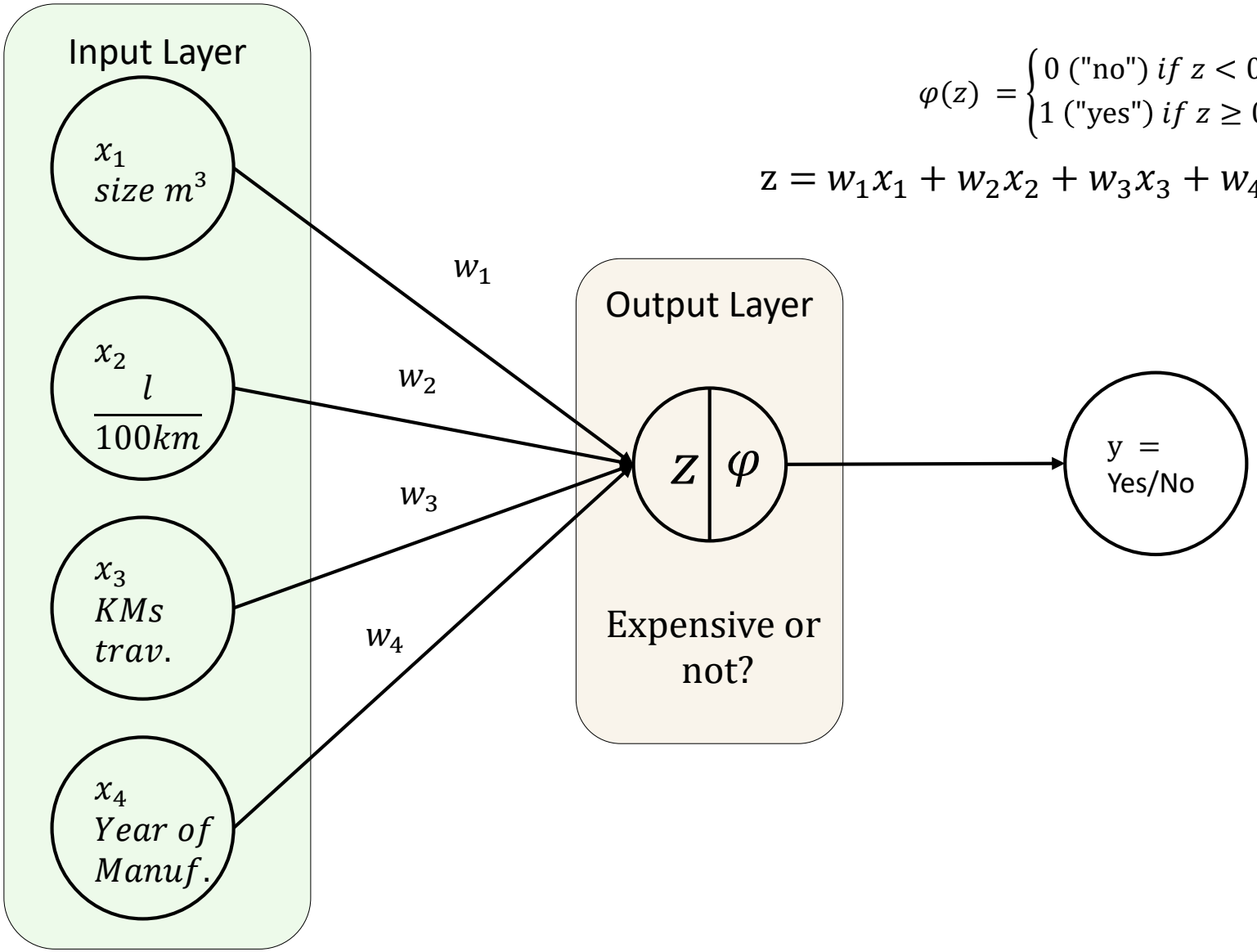




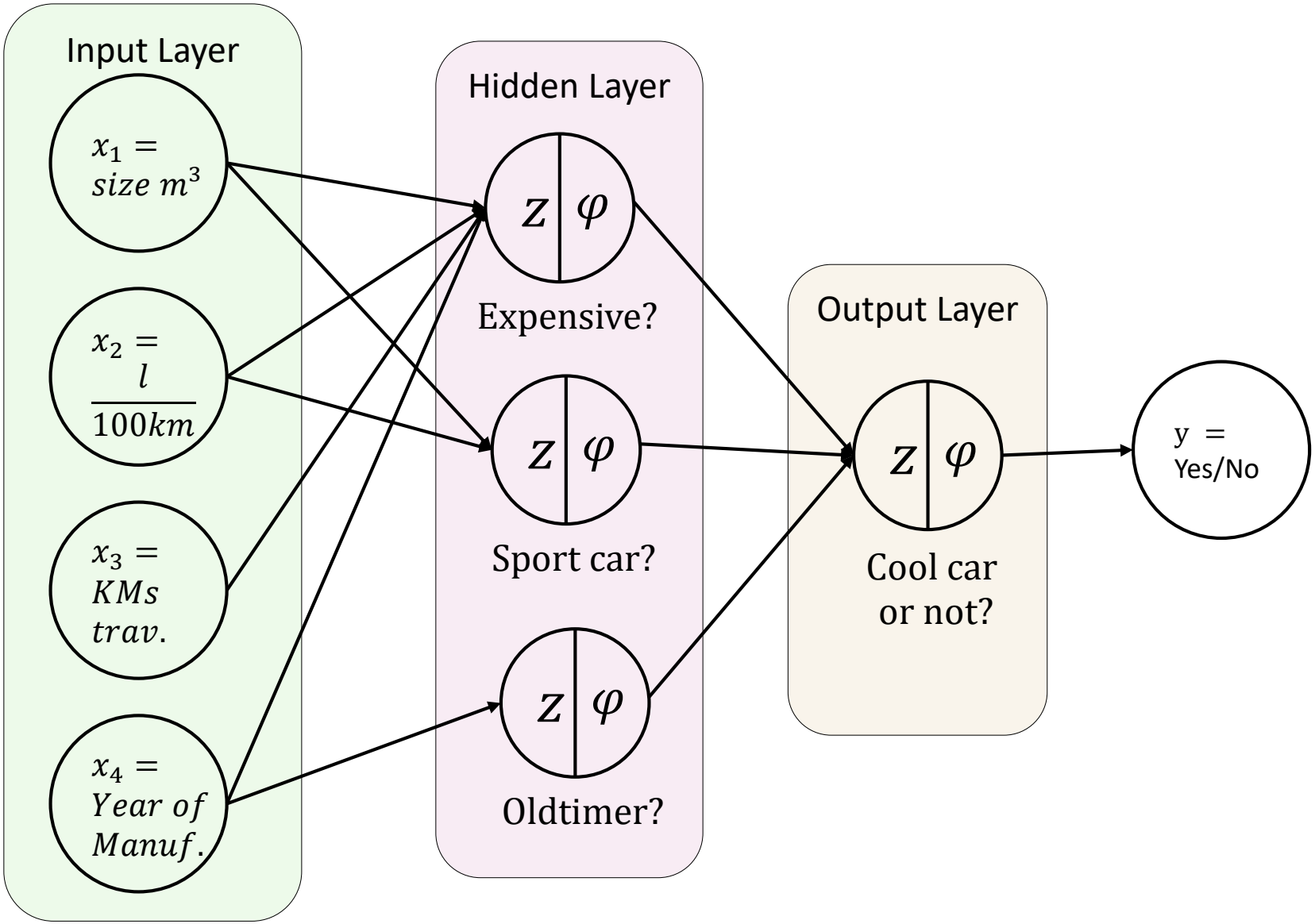


http://neuralnetworksanddeeplearning.com/chap4.html

Intuition: Cool Car or Not? (1)

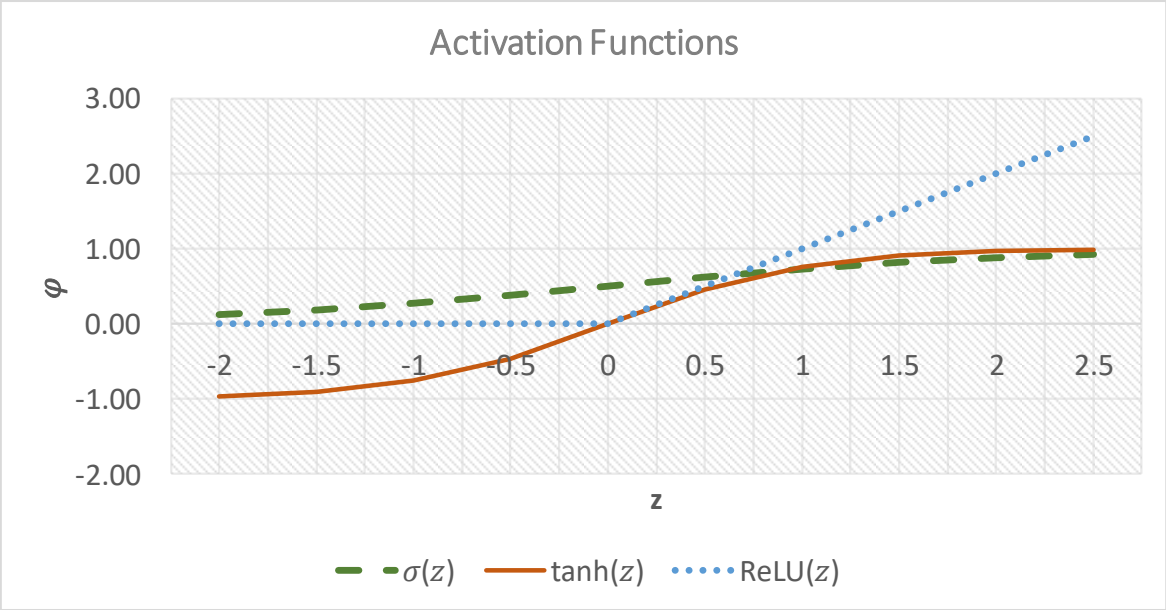


Intuition: Cool Car or Not? (2)



Activation Functions

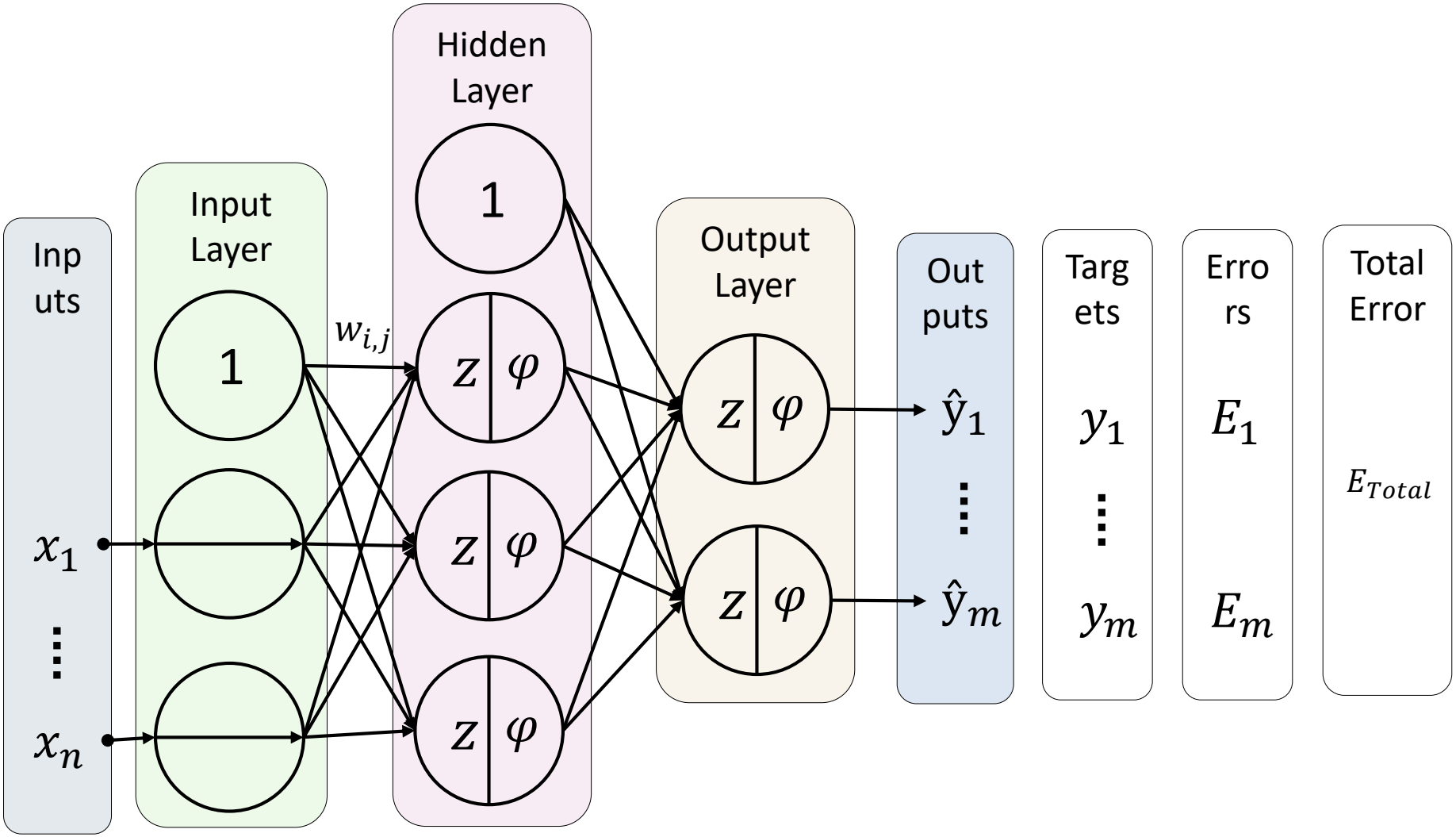
- ~~**Step Activation Function** $h(z)$~~
- **Originally used for MLP: Sigmoid / Logistic Function** $\sigma(z) = \frac{1}{1+e^{-z}}$ **(was most similar to biological neurons)**
- **Hyperbolic Tangent Function** $\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
- **Rectified Linear Unit function** $\mathrm{ReLU}(z) = \max(0, z)$



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Training Multi-Layer NN

Multi-Layer / 1-Hidden Layer Neural Network (Extended Illustration)

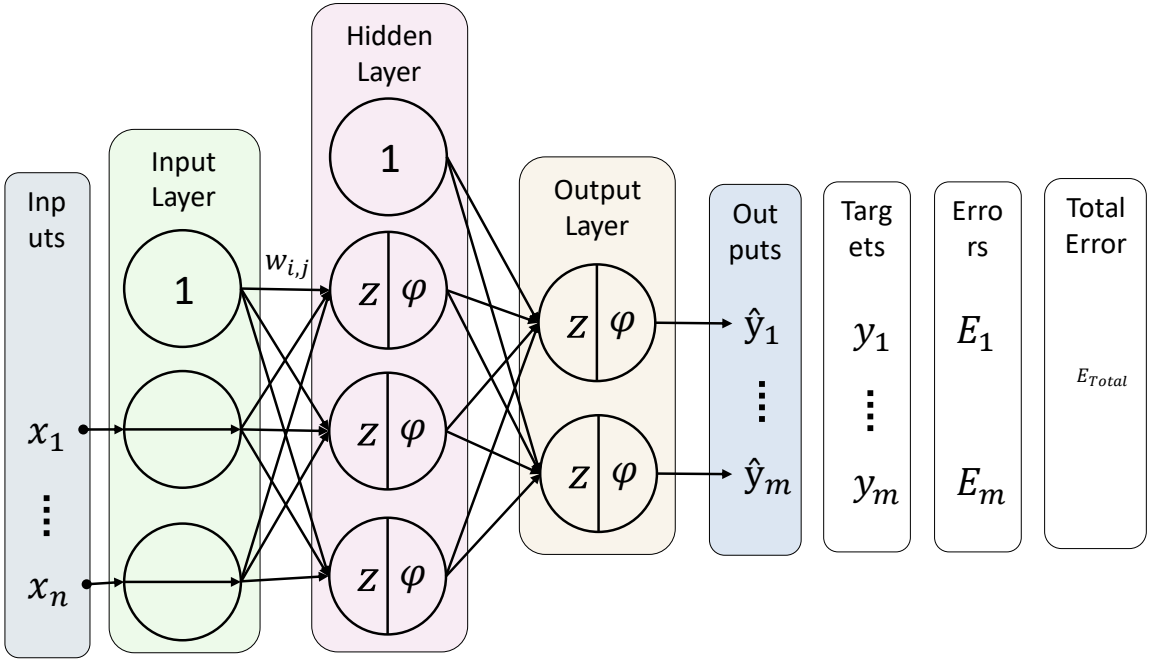


Overview

- **How can we find the optimal weights?**
- **Forward Propagation:**
  - Calculate Outputs
- **Calculate Error(s)**
- **Backpropagation**
  - How much does a change in $w_{j,i}$ affects the cost function, i.e the total error?
  - Adjust weights to minimize cost function

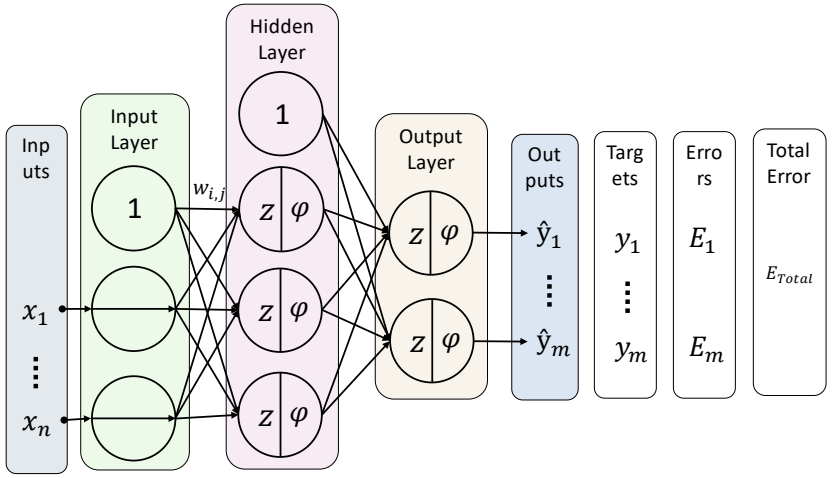
→ The partial derivative of E_{total} with respect to each $w_{j,i}$, i.e. each gradient with respect to $w_{j,i}$

$$\frac{\partial E_{total}}{\partial w_{j,i}}$$



Gradient

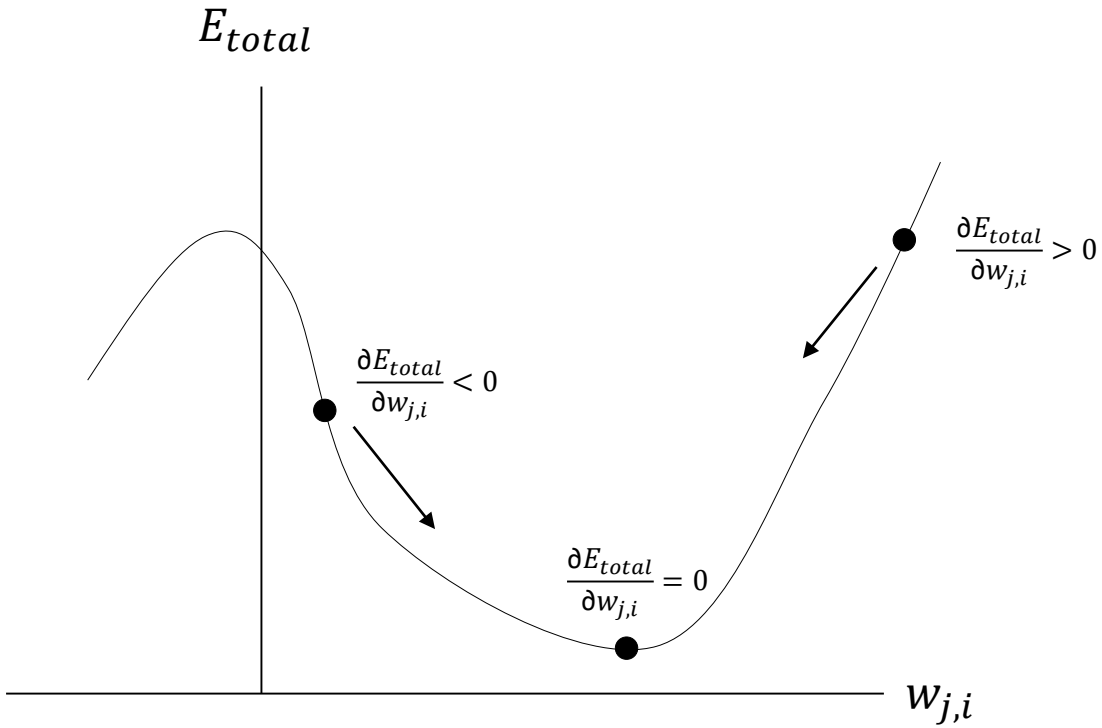
- $E_{total} \geq 0$
- $w_{j,i} \epsilon \mathbb{R}$



Update Rule for MLP

$$w_{j,i} = w_{j,i} - \eta \frac{\partial E_{total}}{\partial w_{j,i}}$$

- Challenge
  - Several Errors
  - Many w's in several layers



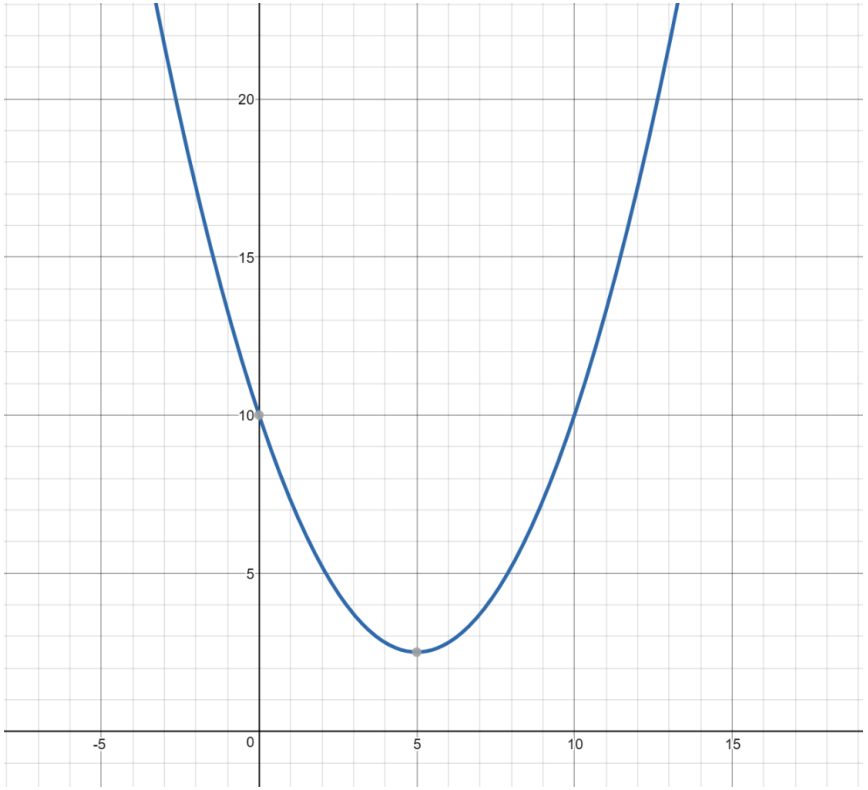
Math Freshup

- **What is the derivative of f(x)?**

E_{total}

$$\mathrm{f}\,(\mathrm{x}) = 0.3x^2 - 3x + 10$$

$$\mathrm{f}'(\mathrm{x}) = 0.6x \quad - 3$$



$w_{j,i}$

Go to **www.menti.com** and use the code **50 13 9**

Who knew what the derivative of f (x) was?

Mentimeter

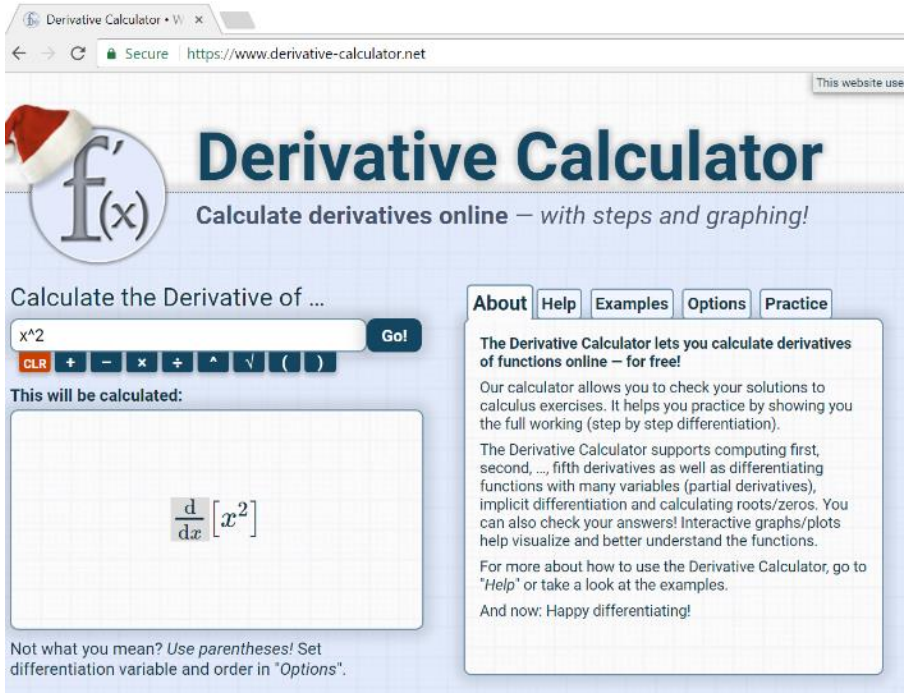
0	0	0
I knew	I think I knew	I didn't know

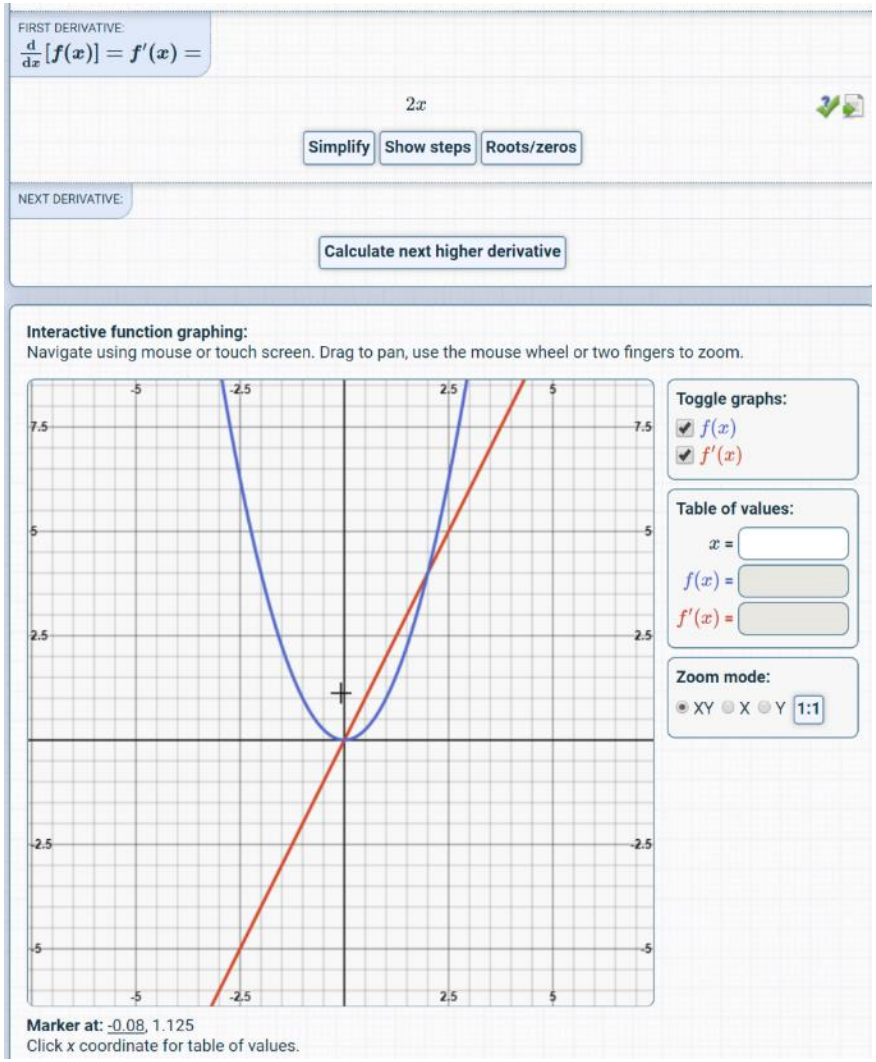
Slide is not active **Activate**

0

Online Calculator

https://www.derivative-calculator.net

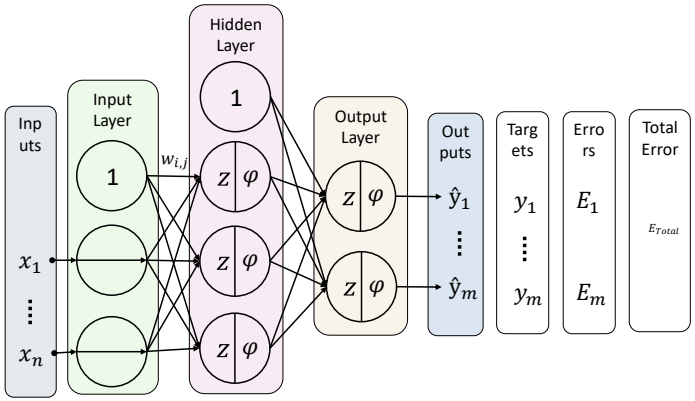


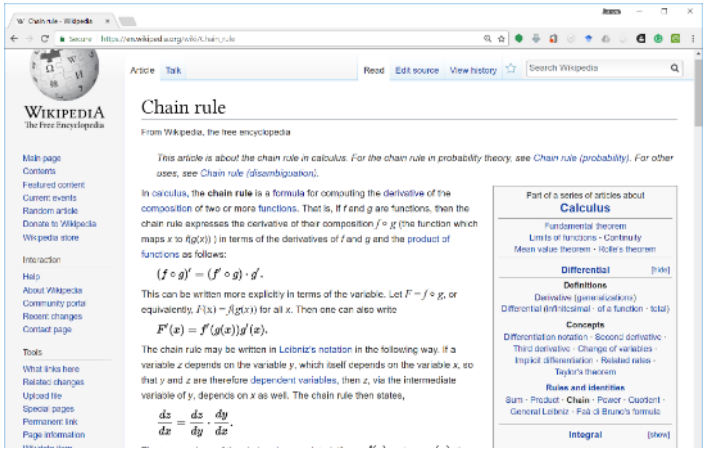


Chain Rule

$$F(x) = (f \circ g)(x) \rightarrow F(x) = f(g(x)) \rightarrow F'(x) = f'\big(g(x)\big) * g'(x)$$

$$y = f(u) \text{ and } u = g(x) \longrightarrow \frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$$



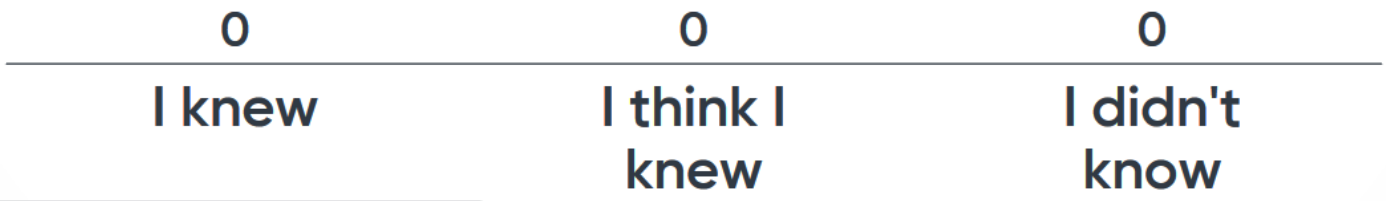


$$z = \sum_{i=0}^{n} w_i x_i \qquad \varphi(z) = \sigma(z) = \frac{1}{1+e^{-z}} \qquad E_{total} = \sum \frac{1}{2}(y - \hat{y})^2$$

Go to **www.menti.com** and use the code **50 13 9**

Who knew the chain rule?

Mentimeter



Slide is not active **Activate**

0

Backpropagation

Backpropagation Illustration

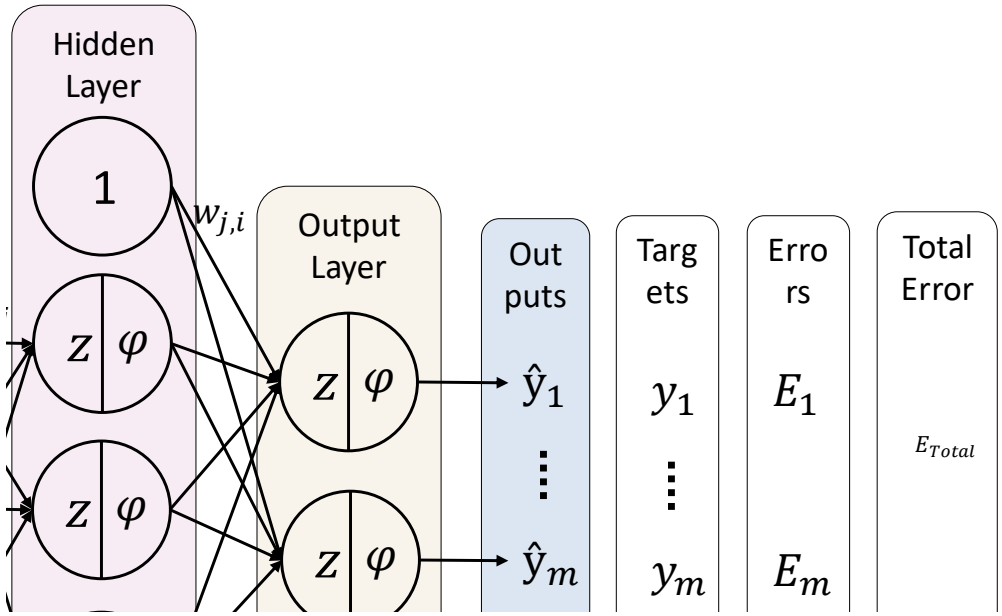
- $\frac{\partial E_{total}}{\partial w_{j,i}} = \frac{\partial E_1}{\partial w_{j,i}} + \ldots + \frac{\partial E_m}{\partial w_{j,i}}$

$$w_{j,i} = w_{j,i} + \eta\delta_j x_{j,i} = w_{j,i} - \eta\frac{\partial E_{total}}{\partial w_{j,i}}$$

- **Via chain-rule**

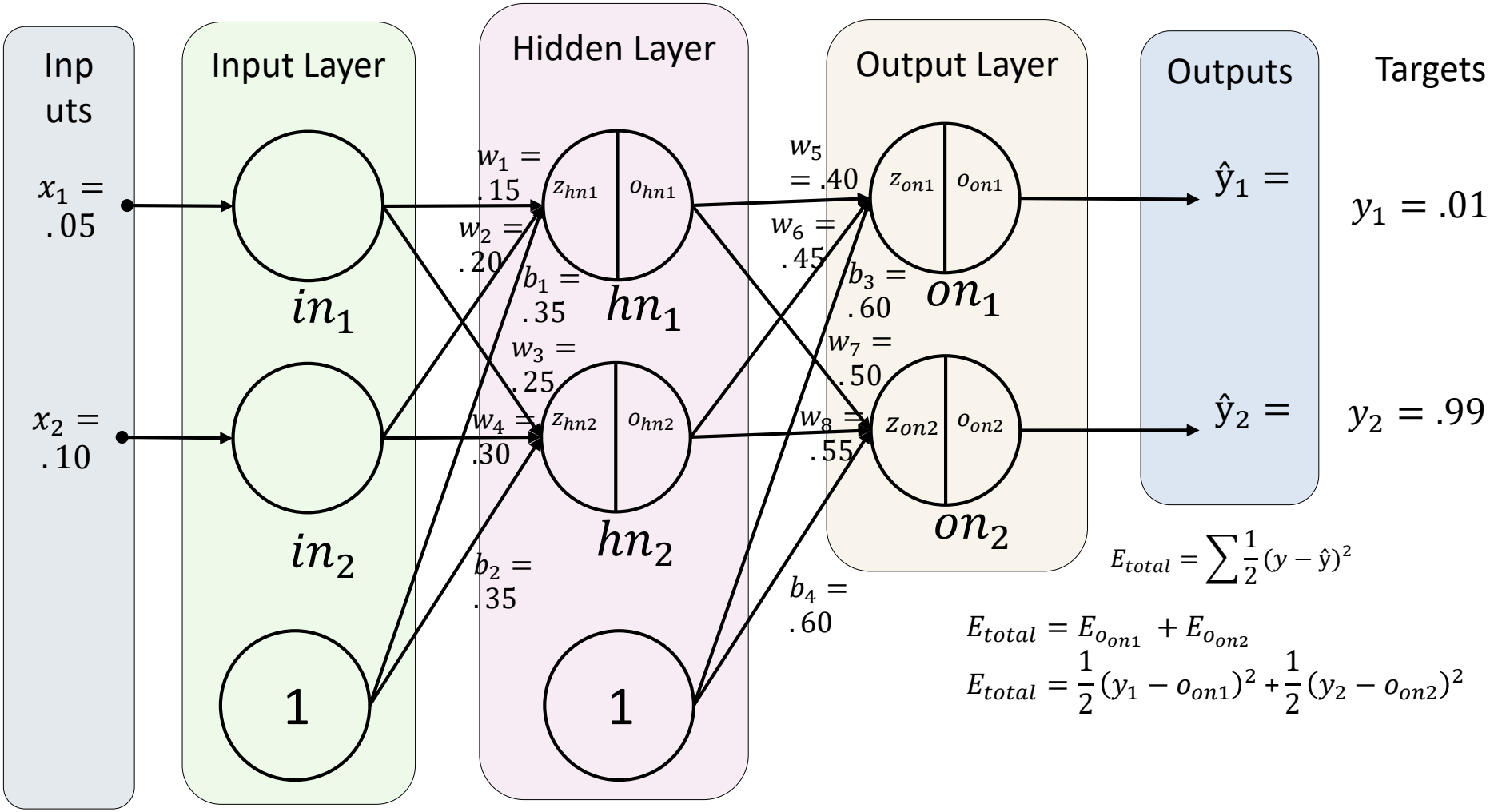
$$\frac{\partial E_1}{\partial w_{j,i}} = \frac{\partial E_1}{\partial \varphi}\frac{\partial \varphi}{\partial z}\frac{\partial z}{\partial w_{j,i}}$$

$$\frac{\partial E_m}{\partial w_{j,i}} = \ldots$$



Example: Initialisation

$$Net\ Input\ z = b_k + \sum_{i=1}^{n} w_i x_i \qquad Neuron's\ Output\ o = \varphi(z) \qquad \varphi(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$



$$E_{total} = \sum \frac{1}{2}(y - \hat{y})^2$$

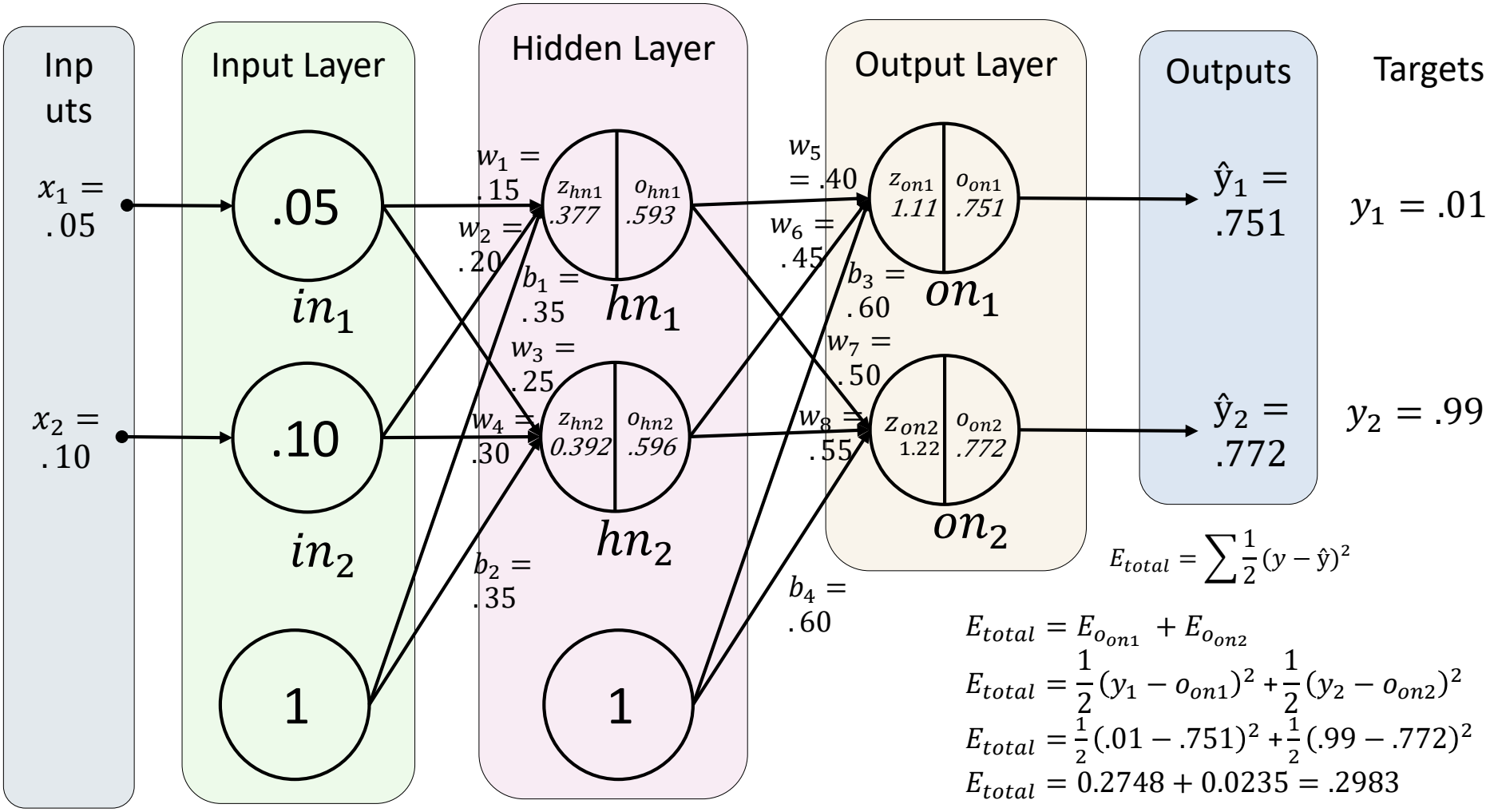
$$E_{total} = E_{o_{on1}} + E_{o_{on2}}$$

$$E_{total} = \frac{1}{2}(y_1 - o_{on1})^2 + \frac{1}{2}(y_2 - o_{on2})^2$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Forward Propagation

$$Net\ Input\ z = b_k + \sum_{i=1}^{n} w_i x_i \qquad Neuron's\ Output\ o = \varphi(z) \qquad \varphi(z) = \sigma(z) = \frac{1}{1+e^{-z}}$$



$$E_{total} = \sum \frac{1}{2}(y - \hat{y})^2$$

$$E_{total} = E_{o_{on1}} + E_{o_{on2}}$$

$$E_{total} = \frac{1}{2}(y_1 - o_{on1})^2 + \frac{1}{2}(y_2 - o_{on2})^2$$

$$E_{total} = \frac{1}{2}(.01 - .751)^2 + \frac{1}{2}(.99 - .772)^2$$

$$E_{total} = 0.2748 + 0.0235 = .2983$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Adjust Weights in Output Layer

$E_{total} = .2983$

$$\frac{\partial E_{total}}{\partial w_5} = \left[\frac{\partial E_{total}}{\partial o_{on1}}\right]\frac{\partial o_{on1}}{\partial z_{on1}}\frac{\partial z_{on1}}{\partial w_5} = \left[\frac{\partial E_1}{\partial o_{on1}} + \frac{\partial E_2}{\partial o_{on1}}\right]\frac{\partial o_{on1}}{\partial z_{on1}}\frac{\partial z_{on1}}{\partial w_5}$$

$$E_{total} = E_1 + E_2 = \left[\frac{1}{2}(y_1 - o_{on1})^2\right] + \left[\frac{1}{2}(y_2 - o_{on2})^2\right]$$

$$\frac{\partial E_{total}}{\partial o_{on1}} = -(y_1 - o_{on1}) + 0 = 0.741$$

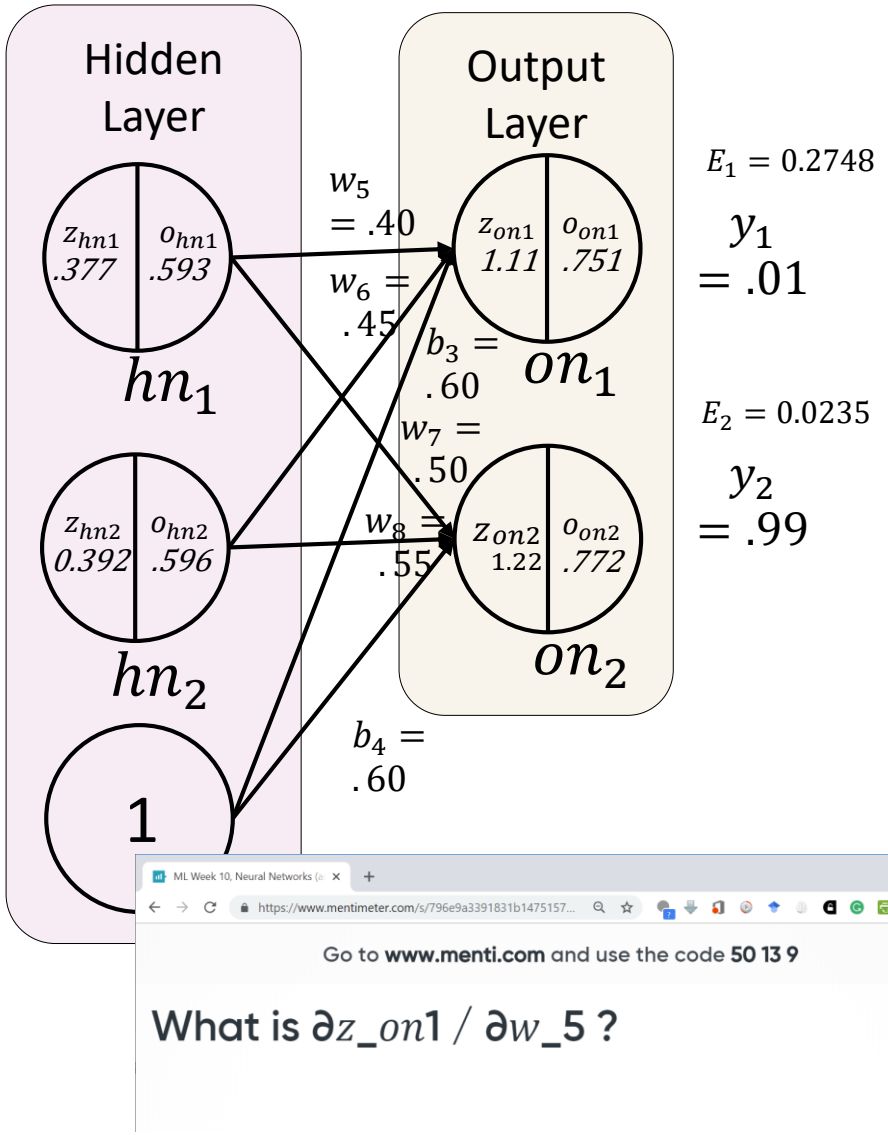
$$o_{on1} = \sigma(z_{on1}) = \frac{1}{1 + e^{-z_{on1}}}$$

$$\frac{\partial o_{on1}}{\partial z_{on1}} = o_{on1}(1 - o_{on1}) = 0.186$$

$$z_{on1} = [w_5 * o_{hn1}] + [w_6 * o_{hn2}] + [b_3]$$

$$\frac{\partial z_{on1}}{\partial w_5} = ?$$

$$\frac{\partial E_{total}}{\partial w_5} =$$



The new Weights

$$w_5 = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$

$$w_5 = 0.40 - 0.50 * 0.082 = 0.358$$

η=0.50

$w_6 = 0.408$

$w_7 = \ldots$

$w_8 = \ldots$

$b_2 = \ldots$

$b_3 = \ldots$

Exam advise

- **If you have to do some calculation in the exam, and solutions are provided as multiple-choice, don‘t get confused by minor differences between your results and the given answers. Differences may occur due to rounding. For instance, if the following answers were given…**

1. 0.43
2. 0.23
3. 0.77
4. 0.261
5. None of the above

- **… and you calculated 0.78, then answer 3 would be the right choice (unless you did something completely wrong and another answer is correct, of course).**

Derivative of Sigmoid

$$o_{on1} = \sigma(z_{on1}) = \frac{1}{1 + e^{-z_{on1}}}$$

$$\rightarrow \sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] = \frac{d}{dx}[(1 + e^{-x})^{-1}] = -(1 + e^{-x})^{-2}(-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} * \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} * \frac{(1+e^{-x})-1}{1+e^{-x}} = \frac{1}{1+e^{-x}} * \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = \frac{1}{1+e^{-x}} * \left(1 - \frac{1}{1+e^{-x}}\right) = \sigma(x) * (1 - \sigma(x))$$

Adjust Weights in Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \left[\frac{\partial E_{total}}{\partial o_{hn1}}\right]\frac{\partial o_{hn1}}{\partial z_{hn1}}\frac{\partial z_{hn1}}{\partial w_1} = \left[\frac{\partial E_{on1}}{\partial o_{hn1}} + \frac{\partial E_{on2}}{\partial o_{hn1}}\right]\frac{\partial o_{hn1}}{\partial z_{hn1}}\frac{\partial z_{hn1}}{\partial w_1}$$

$$\frac{\partial E_{on1}}{\partial o_{hn1}} = \left[\frac{\partial E_{on1}}{\partial z_{hn1}}\right] * \left[\frac{\partial z_{on1}}{\partial o_{hn1}}\right] = \left[\frac{\partial E_{on1}}{\partial o_{on1}} * \frac{\partial o_{on1}}{\partial z_{on1}}\right] * [w_5] = [0.741 * 0.186] * [0.4] = 0.055$$

$$\frac{\partial E_{on2}}{\partial o_{hn1}} = -0.019$$

$$\frac{\partial E_{total}}{\partial o_{on1}} = 0.055 + (-0.019) = 0.036$$

$$o_{on1} = \sigma(z_{on1}) = \frac{1}{1 + e^{-z_{on1}}}$$

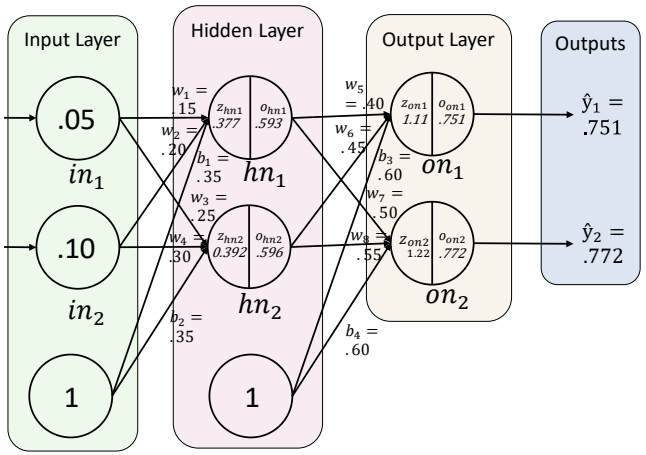
$$\frac{\partial o_{on1}}{\partial z_{on1}} = o_{hn1}(1 - o_{hn1}) = 0.241$$

$$z_{on1} = [w_1 * o_{in1}] + [w_2 * o_{in2}] + [b_1]$$

$$\frac{\partial z_{on1}}{\partial w_1} = [1 * o_{in1}] + [0] + [0] = 0.05$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.036 * 0.241 * 0.05 = 0.00043$$

$E_{total} = .2983$



How often?

- **First iteration**
  - $E_{total} = .298$
  - $\hat{y}_1 = .751 \qquad y_1 = .01$
  - $\hat{y}_2 = .751 \qquad y_2 = .99$
- **Second iteration**
  - $E_{total} = .291$
- **After 10,000 iterations:**
  - $E_{total} = .0000351$
  - $\hat{y}_1 = .0159 \qquad y_1 = .01$
  - $\hat{y}_2 = .984 \qquad y_2 = .99$

Backpropagation Algorithm / Delta Training Rule

REPEAT UNTIL Termination Condition is Met

FOR EACH Instance in Training Dataset

DO Propagate Forward, i.e. Calculate Outputs

DO Propagate Errors Back Through Network

FOR EACH Output Neuron

DO Calculate Error Term

$$\delta_i^O = \hat{y}_i \left(1 - \hat{y}_i\right)\left(y_i - \hat{y}_i\right)$$

FOR EACH Hidden Neuron

$y_i = target\ output$
$\hat{y}_i = actual\ output\ of\ ith\ neuron\ (output\ layer)$
$\hat{y}_j = output\ of\ jth\ neuron\ (hidden\ layer)$
$x_{j,i} = output\ from$ previous' layer neuron

DO Calculate Error Term

$$\delta_j^H = \hat{y}_j\ (1 - \hat{y}_j\ ) \sum_i \delta_i^O w_{j,i}$$

DO Update Weights

$$w_{j,i} = w_{j,i} + \Delta w_{j,i} = w_{j,i} + \eta \delta_j x_{j,i} = w_{j,i} - \eta \frac{\partial E_{total}}{\partial w_{j,i}}$$

http://www.cogsys.wiai.uni-bamberg.de/teaching/ss05/ml/slides/cogsysII-4.pdf

Backpropagation Algorithm / Delta Training Rule (Again)

Table 5.2 *Backpropagation of error* in a neural network with one hidden layer

1. Present example $\mathbf{x}$ to the input layer and propagate it through the network.
2. Let $\mathbf{y} = (y_1, \ldots y_m)$ be the output vector, and let $\mathbf{t}(\mathbf{x}) = (t_1, \ldots t_m)$ be the target vector.
3. For each output neuron, calculate its responsibility, $\delta_i^{(1)}$, for the network's error:
   $\delta_i^{(1)} = y_i(1 - y_i)(t_i - y_i)$
4. For each hidden neuron, calculate its responsibility, $\delta_j^{(2)}$, for the network's error. While doing so, use the responsibilities, $\delta_i^{(1)}$, of the output neurons as obtained in the previous step.
   $\delta_j^{(2)} = h_j(1 - h_j) \sum_i \delta_i^{(1)} w_{ji}$
5. Update the weights using the following formulas, where η is the learning rate:

   output layer: $w_{ji}^{(1)} := w_{ji}^{(1)} + \eta \delta_i^{(1)} h_j$; h_j: the output of the j-th hidden neuron
   hidden layer: $w_{kj}^{(2)} := w_{kj}^{(2)} + \eta \delta_j^{(2)} x_k$; x_k: the value of the k-th attribute

6. Unless a termination criterion has been satisfied, return to step 1.

Miroslav Kubat, An Introduction to Machine Learning (Springer, 2015).

Costs of Backpropagation, Example

- **100 input features/neurons**
- **100 hidden neurons**
- **→ 10^4 weights → 10^4 changes after each training instance**
- **Upper layer weights can be neglected (e.g. 3 classes → 300 weights)**
- **10^5 training instances**
- **10^4 epochs**

- **→ 10^4 * 10^5 *10^4 = 10^13 weight updates**

Miroslav Kubat, An Introduction to Machine Learning (Springer, 2015).

Vanishing Gradient Problem

- **The more hidden layers, the less well the standard backpropagation algorithm works**
- **Possible Solutions**
  - Multi-Level Hierachy
  - Long-term short memory

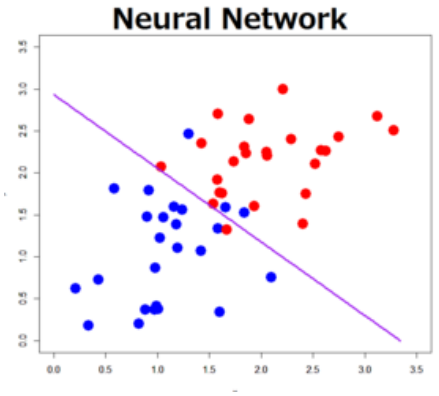
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

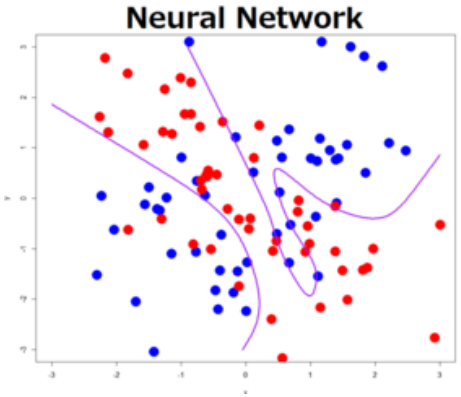
Practical Issues

„Designing Neural Networks is more Art than Science“

Number of hidden layers

- **0 Hidden Layers (i.e. 1 layer of LTUs) = linearly separable function**
- **1 Hidden Layer = Any function for continuous mapping from one finite space to another ← often enough (hence no more research for a while)**
- **2 Hidden Layers = Any decision boundary ← "There is currently no theoretical reason to use neural networks with any more than two hidden layers." (again, no more research for a while)**
- **More layers**
  - Intuitive: Model from abstract to complex
  - potentially higher parameter efficiency (exponentially fewer neurons required) → faster training
  - Potentially less effective backpropagation / more unstable gradient descent /more local minima
  - Example
  - 97% accuracy on MNIST dataset with one hidden layer
  - 98% accuracy with 2 hidden layers



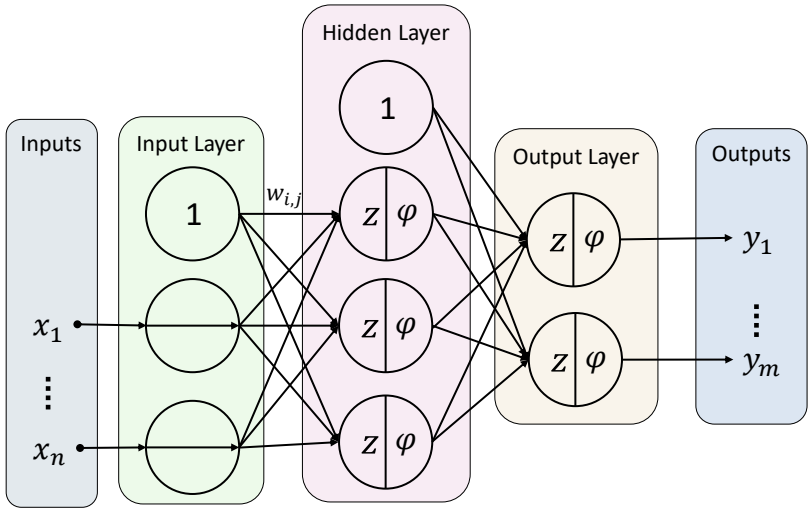


Aurélien Géron, Hands on Machine Learning with scikit-learn and Tensorflow (O'Reilly Media, 2017).
Timothy Masters. Practical Neural Network Recipies in C++. 1993
Jeff Heaton. Introduction to Neural Networks with Java

Number of Neurons

- **Number of Input Neurons** $|N^x| = Number\ of\ features$
- **Number of Output Neurons** $|N^y| = Number\ of\ (dummy)\ classes$ (or 1)
- **Number of Hidden Neurons** $|N^{hid}|$
  - Too many: overfitting and long training time
  - Too few: underfitting
  - $|N^{hid}| = \frac{|D|}{\alpha(N^x+N^y)}$
  - $|N^{hid}| = \frac{2}{3}(N^x + N^y)$
  - $|N^{hid}| = \sqrt{N^x N^y}$

$|D| = Number\ of\ Instances\ in\ training\ data$
$\alpha = [5 \dots 10]\ (abitrarily\ selected\ scaling\ factor)$

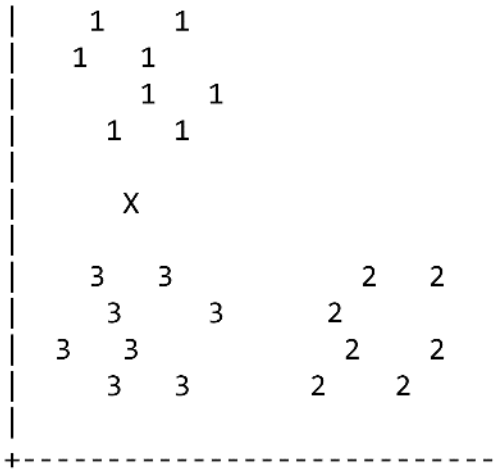


Which activation function to use

- **ReLU for hidden layers**
  - Fast to compute
  - Gradient Descent does not get stuck on plateaus
- **Softmax for output layer (if classes are mutually exclusive)**
- **No activation function for regression**
- **Different functions within one layer?**
  - Theoretically yes
  - Practically no (to the best of my knowledge)

Feature Engineering

- **Categorical Input Variables → Encode as Dummy Variables**
- **Categorical Output → Use Softmax**
- **Scaling**
  - In theory not necessary (weights compensate)
  - In practice, there are advantages
    - Faster training
    - More robust (same effect of learning rate etc.)



ftp://ftp.sas.com/pub/neural/FAQ2.html

- **More Details: ftp://ftp.sas.com/pub/neural/FAQ2.html#A_std_in**

Data Type	Action
Numeric x	Standardization or Normalization
Binary x	-1 / +1 encoding
Categorical x (no weight decay)	1-of-(C-1) dummy encoding
Categorical x (weight decay)	1-of-C dummy encoding
Numeric y	As it is
Categorical y	1-of-C dummy encoding with Softmax

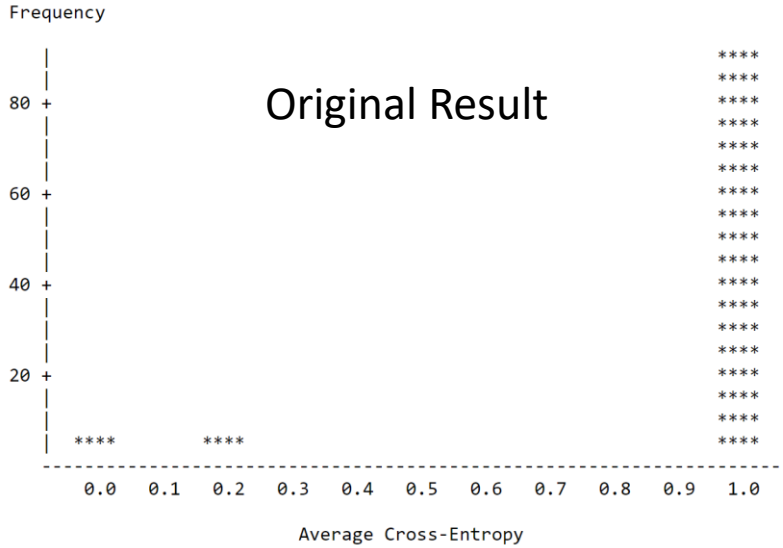
https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx

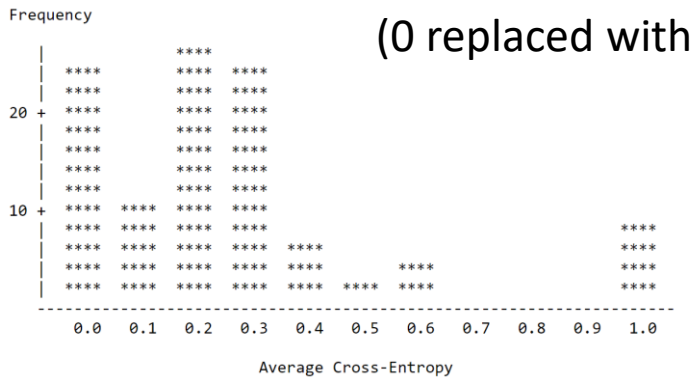
Binary Inputs: -1 or 0? (I)

One hundred networks were trained from different random initial weights. The following bar chart shows the distribution of the average cross-entropy after training:

Input

x1	x2	x3	x4	x5	target
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	1
1	1	0	0	0	0
0	0	1	0	0	1
1	0	1	0	0	0
0	1	1	0	0	0
1	1	1	0	0	1
0	0	0	1	0	1
1	0	0	1	0	0
0	1	0	1	0	0
1	1	0	1	0	1
0	0	1	1	0	0
1	0	1	1	0	1
0	1	1	1	0	1
1	1	1	1	0	0
0	0	0	0	1	1
1	0	0	0	1	0
0	1	0	0	1	0
1	1	0	0	1	1
0	0	1	0	1	0
1	0	1	0	1	1
0	1	1	0	1	1
1	1	1	0	1	0
0	0	0	1	1	0
1	0	0	1	1	1
0	1	0	1	1	1
1	1	0	1	1	0
0	0	1	1	1	1
1	0	1	1	1	0
0	1	1	1	1	0
1	1	1	1	1	1



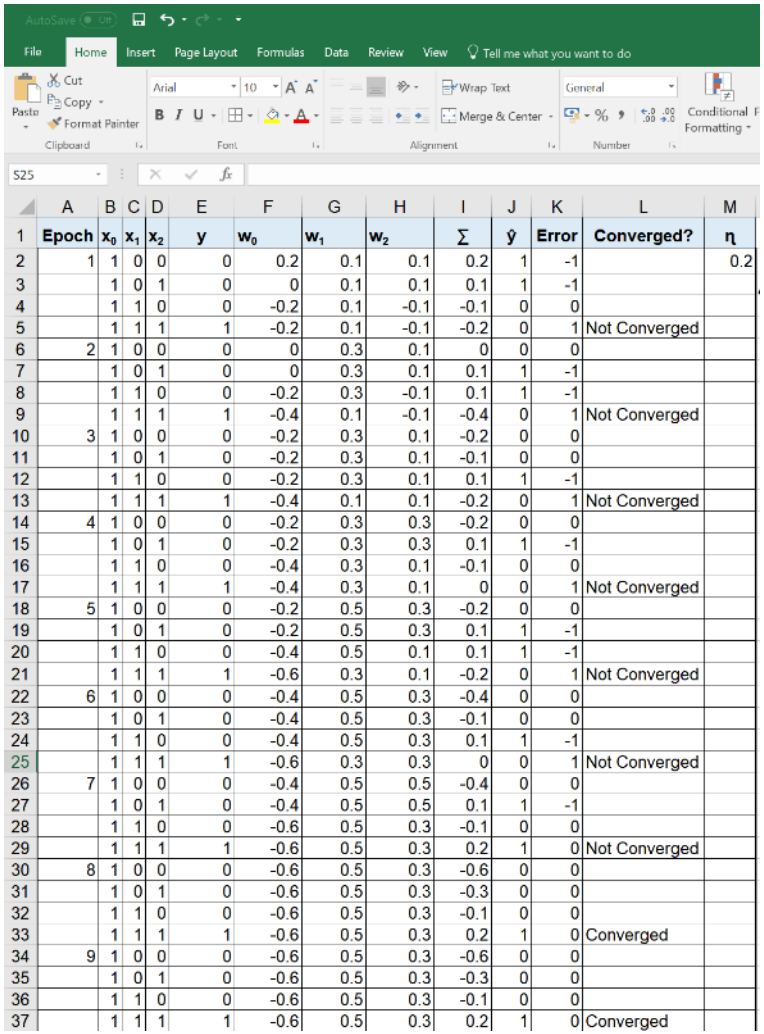


ftp://ftp.sas.com/pub/neural/FAQ2.html#A_binary

Binary Inputs: -1 or 0? (II)

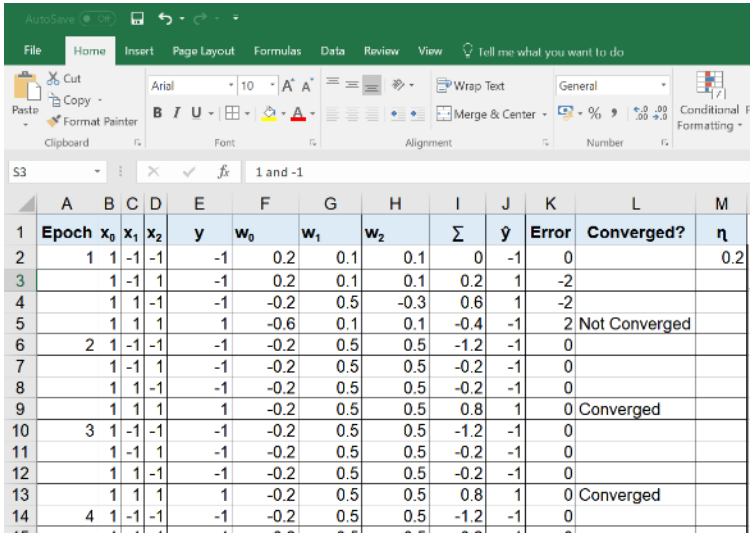
AND Operation

With 0



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Epoch	x_0	x_1	x_2	y	w_0	w_1	w_2	Σ	ŷ	Error	Converged?	η
2	1	1	0	0	0	0.2	0.1	0.1	0.2	1	-1		0.2
3		1	0	1	0	0	0.1	0.1	0.1	1	-1		
4		1	1	0	0	-0.2	0.1	-0.1	-0.1	0	0		
5		1	1	1	1	-0.2	0.1	-0.1	-0.2	0	1	Not Converged	
6	2	1	0	0	0	0	0.3	0.1	0	0	0		
7		1	0	1	0	0	0.3	0.1	0.1	1	-1		
8		1	1	0	0	-0.2	0.3	-0.1	0.1	1	-1		
9		1	1	1	1	-0.4	0.1	-0.1	-0.4	0	1	Not Converged	
10	3	1	0	0	0	-0.2	0.3	0.1	-0.2	0	0		
11		1	0	1	0	-0.2	0.3	0.1	-0.1	0	0		
12		1	1	0	0	-0.2	0.3	0.1	0.1	1	-1		
13		1	1	1	1	-0.4	0.1	0.1	-0.2	0	1	Not Converged	
14	4	1	0	0	0	-0.2	0.3	0.3	-0.2	0	0		
15		1	0	1	0	-0.2	0.3	0.3	0.1	1	-1		
16		1	1	0	0	-0.4	0.3	0.1	-0.1	0	0		
17		1	1	1	1	-0.4	0.3	0.1	0	0	1	Not Converged	
18	5	1	0	0	0	-0.2	0.5	0.3	-0.2	0	0		
19		1	0	1	0	-0.2	0.5	0.3	0.1	1	-1		
20		1	1	0	0	-0.4	0.5	0.1	0.1	1	-1		
21		1	1	1	1	-0.6	0.3	0.1	-0.2	0	1	Not Converged	
22	6	1	0	0	0	-0.4	0.5	0.3	-0.4	0	0		
23		1	0	1	0	-0.4	0.5	0.3	-0.1	0	0		
24		1	1	0	0	-0.4	0.5	0.3	0.1	1	-1		
25		1	1	1	1	-0.6	0.3	0.3	0	0	1	Not Converged	
26	7	1	0	0	0	-0.4	0.5	0.5	-0.4	0	0		
27		1	0	1	0	-0.4	0.5	0.5	0.1	1	-1		
28		1	1	0	0	-0.6	0.5	0.3	-0.1	0	0		
29		1	1	1	1	-0.6	0.5	0.3	0.2	1	0	Not Converged	
30	8	1	0	0	0	-0.6	0.5	0.3	-0.6	0	0		
31		1	0	1	0	-0.6	0.5	0.3	-0.3	0	0		
32		1	1	0	0	-0.6	0.5	0.3	-0.1	0	0		
33		1	1	1	1	-0.6	0.5	0.3	0.2	1	0	Converged	
34	9	1	0	0	0	-0.6	0.5	0.3	-0.6	0	0		
35		1	0	1	0	-0.6	0.5	0.3	-0.3	0	0		
36		1	1	0	0	-0.6	0.5	0.3	-0.1	0	0		
37		1	1	1	1	-0.6	0.5	0.3	0.2	1	0	Converged	

With -1



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Epoch	x_0	x_1	x_2	y	w_0	w_1	w_2	Σ	ŷ	Error	Converged?	η
2	1	1	-1	-1	-1	0.2	0.1	0.1	0	-1	0		0.2
3		1	-1	1	-1	0.2	0.1	0.1	0.2	1	-2		
4		1	1	-1	-1	-0.2	0.5	-0.3	0.6	1	-2		
5		1	1	1	1	-0.6	0.1	0.1	-0.4	-1	2	Not Converged	
6	2	1	-1	-1	-1	-0.2	0.5	0.5	-1.2	-1	0		
7		1	-1	1	-1	-0.2	0.5	0.5	-0.2	-1	0		
8		1	1	-1	-1	-0.2	0.5	0.5	-0.2	-1	0		
9		1	1	1	1	-0.2	0.5	0.5	0.8	1	0	Converged	
10	3	1	-1	-1	-1	-0.2	0.5	0.5	-1.2	-1	0		
11		1	-1	1	-1	-0.2	0.5	0.5	-0.2	-1	0		
12		1	1	-1	-1	-0.2	0.5	0.5	-0.2	-1	0		
13		1	1	1	1	-0.2	0.5	0.5	0.8	1	0	Converged	
14	4	1	-1	-1	-1	-0.2	0.5	0.5	-1.2	-1	0		

ftp://ftp.sas.com/pub/neural/FAQ2.html#A_binary

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

The Role of TF-IDF for Machine-Learning

Vector Space Model and TF-IDF

- **Each document is represented as a term-vector in an n-dimensional space (n = number of different terms)**
- **Term weights are, for instance, based on term frequency**
- **Document similarity is expressed as the (cosine) distance between two documents/vectors**
- **More advanced term-weighting: TF-IDF**

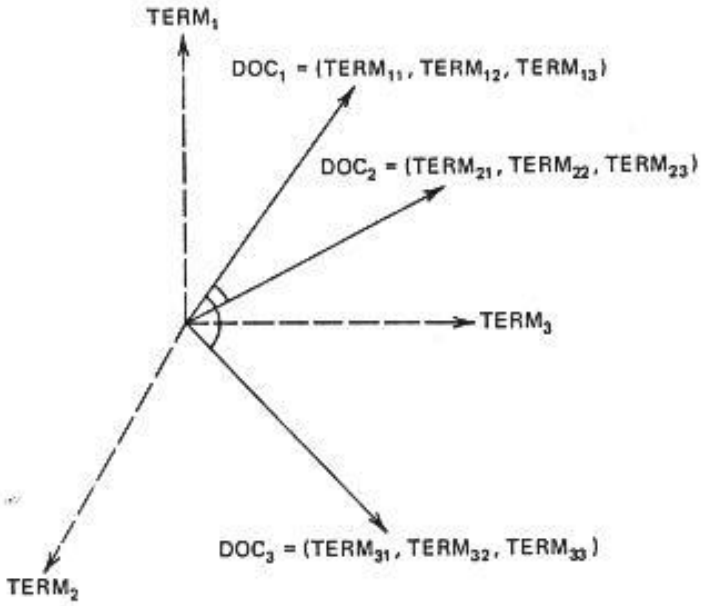


Figure 4-2 Vector representation of document space.

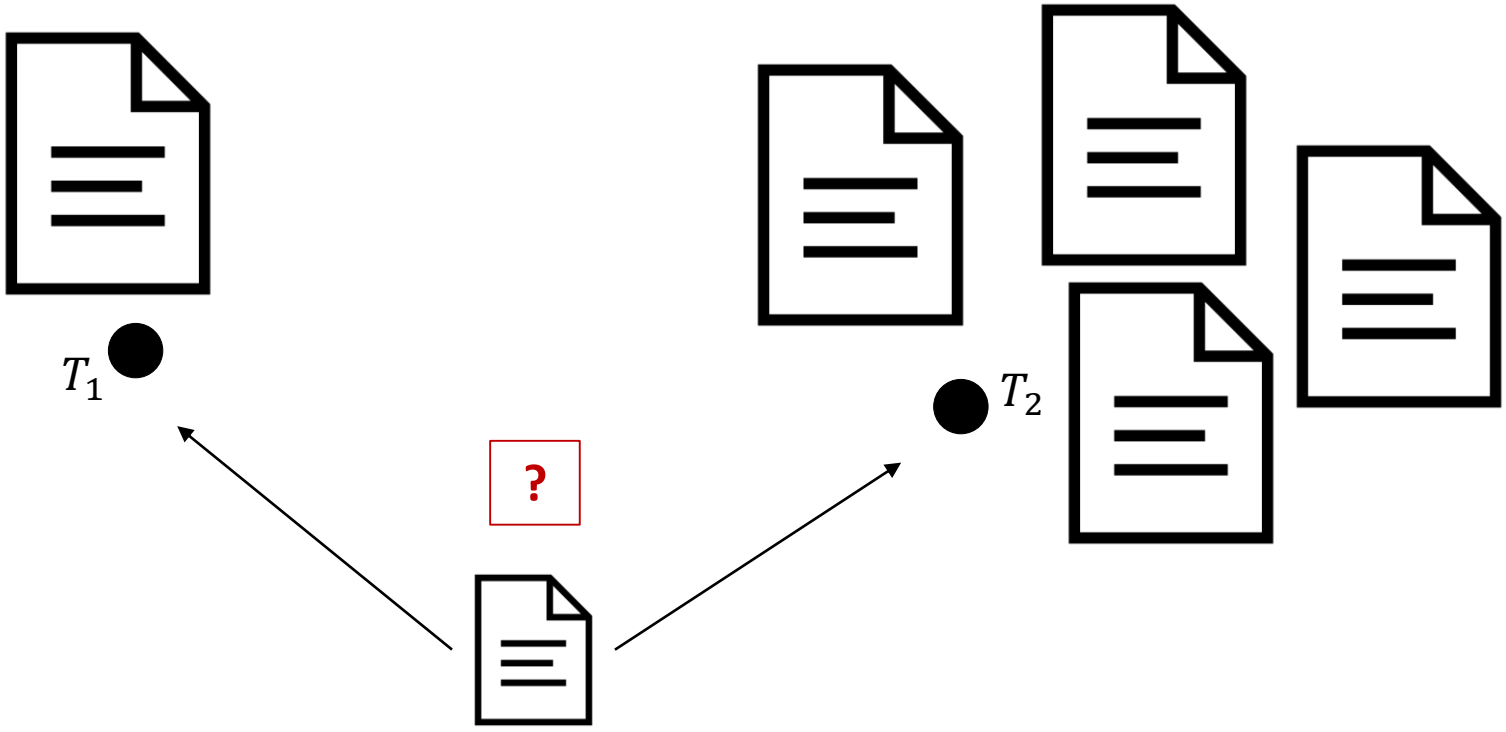
$$TF - IDF = tf(t) * \log \frac{N_r}{n_r}$$

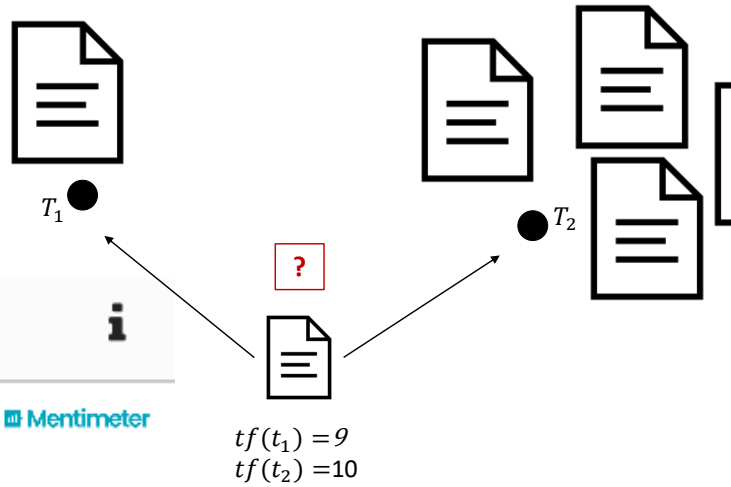
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

https://en.wikipedia.org/wiki/Cosine_similarity
http://www.cs.uni.edu/~okane/source/ISR/vspace.jpg

Document Clustering/Classification

- **Where should the new document go to? Topic T_1 or T_2?**

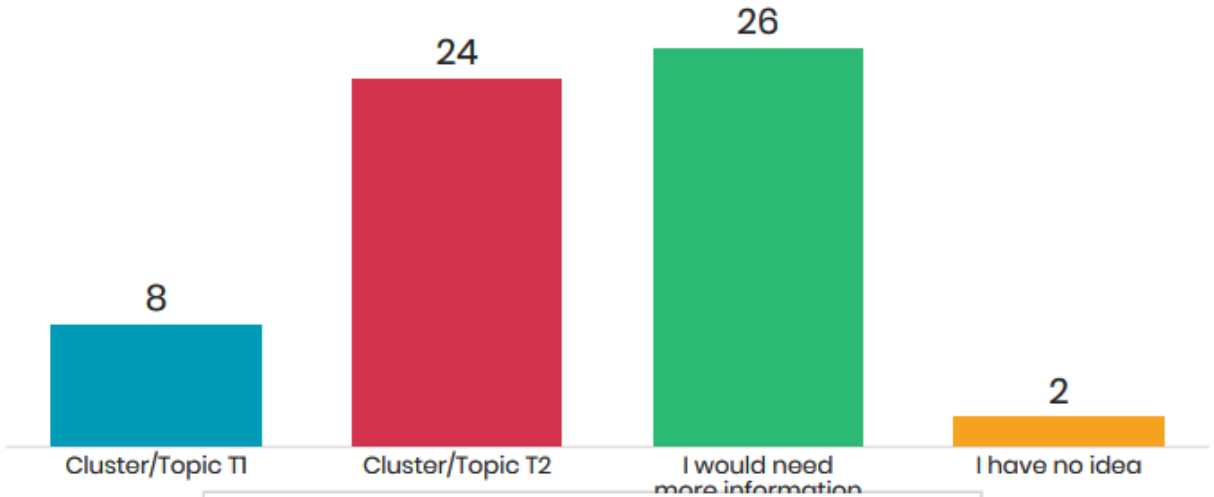




Go to **www.menti.com** and use the code **16 62 1**

To which cluster does the new document belong?

Mentimeter

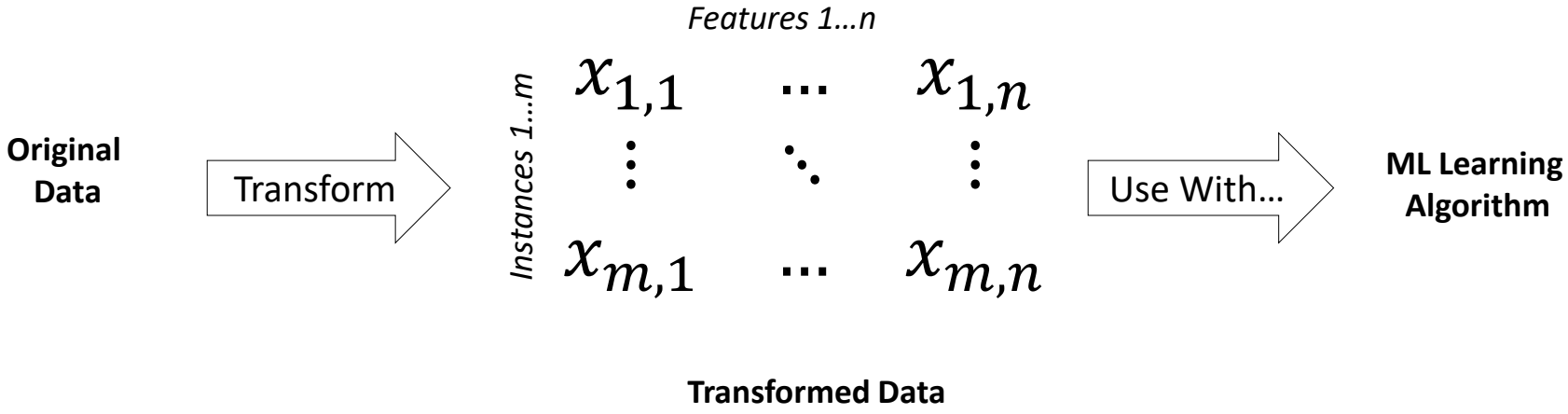


Slide is not active

Activate

60

Illustration Typical ML Workflow

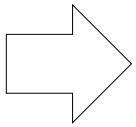


Example: Typical data transformation

- **„Nationality“ probably encoded as dummy variable**
- **„Gender“ probably ignored**
- **Numbers probably standardized/normalized (not in the example)**

Student ID	Age	Nationality	Gender	Hours Spent Assignment 1	Mark Exam
1	21	Irish	Female	96	37
2	21	German	Female	68	38
3	23	Irish	Female	29	57
4	24	Irish	Female	88	49
5	23	Irish	Female	96	78

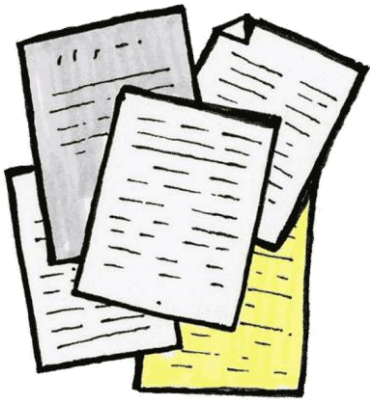
Original Data



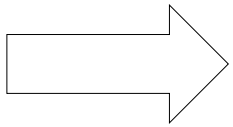
$$\begin{matrix} 21 & 1 & 0 & 96 & 37 \\ 21 & 0 & 1 & 68 & 38 \\ 23 & 1 & 0 & 29 & 57 \\ 24 & 1 & 0 & 88 & 49 \\ 23 & 1 & 0 & 96 & 78 \end{matrix}$$

Transformed Data

How about documents?



Original Data



?

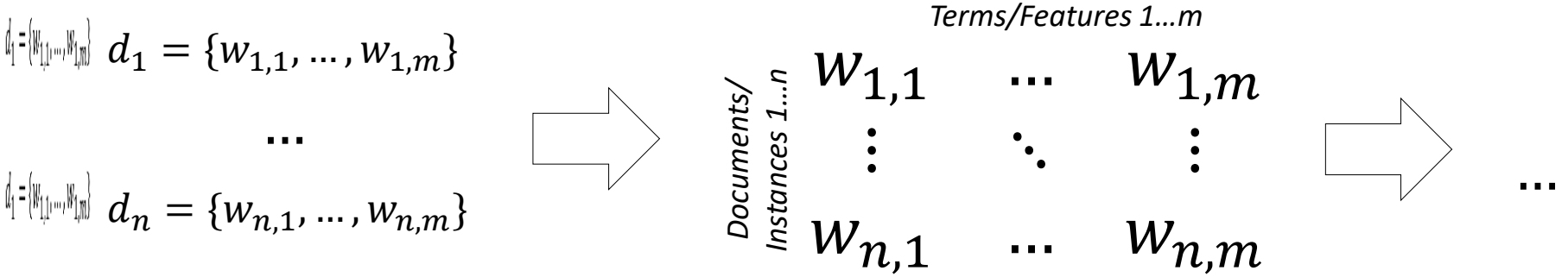
$$\begin{matrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{matrix}$$

Transformed Data

https://bcx.basecamp-static.com/assets/blank_slates/blank_slate_icon_documents@2x-f4bf668018b3b455542d7ea30eda71a3.png

Document-Term Frequency Matrix

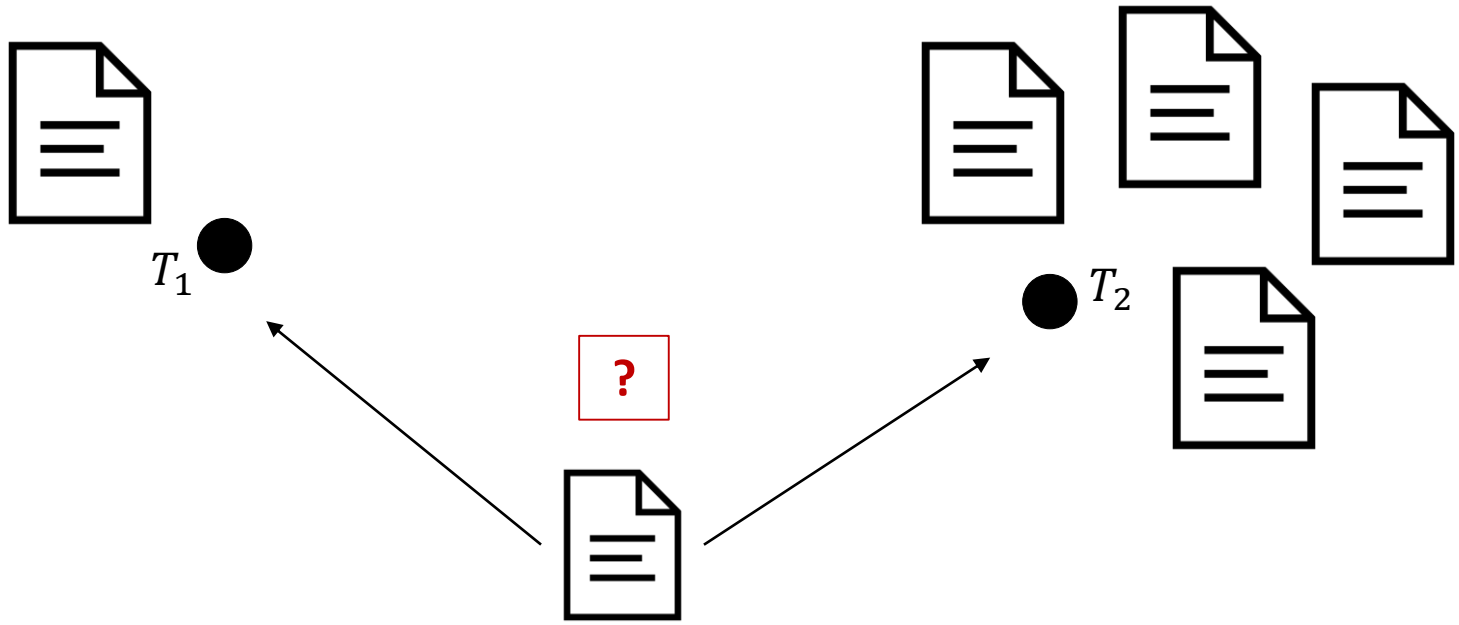
- **In the corpus *D* with *n* documents $d_{i=1\dots n}$ and a vocabulary of *m* terms $t_{j=1\dots m}$, each document d_i is represented as vector of term-document weights $d_i = \{w_{i,1}, \dots, w_{i,m}\}$**
- **How to find the weights?**
  - Most simple: boolean
  - Slightly better: term frequency *tf*
    $w_{i,j} = tf(d_i, t_j)$
  - Usually best: TF-IDF $w_{i,j} = tfidf(d_i, t_j, D)$



Document Clustering/Classification: Boolean

- **What would be the weights?**
- **Where should the new document go to? Topic T_1 or T_2?**

→ With boolean weighting, no decision possible

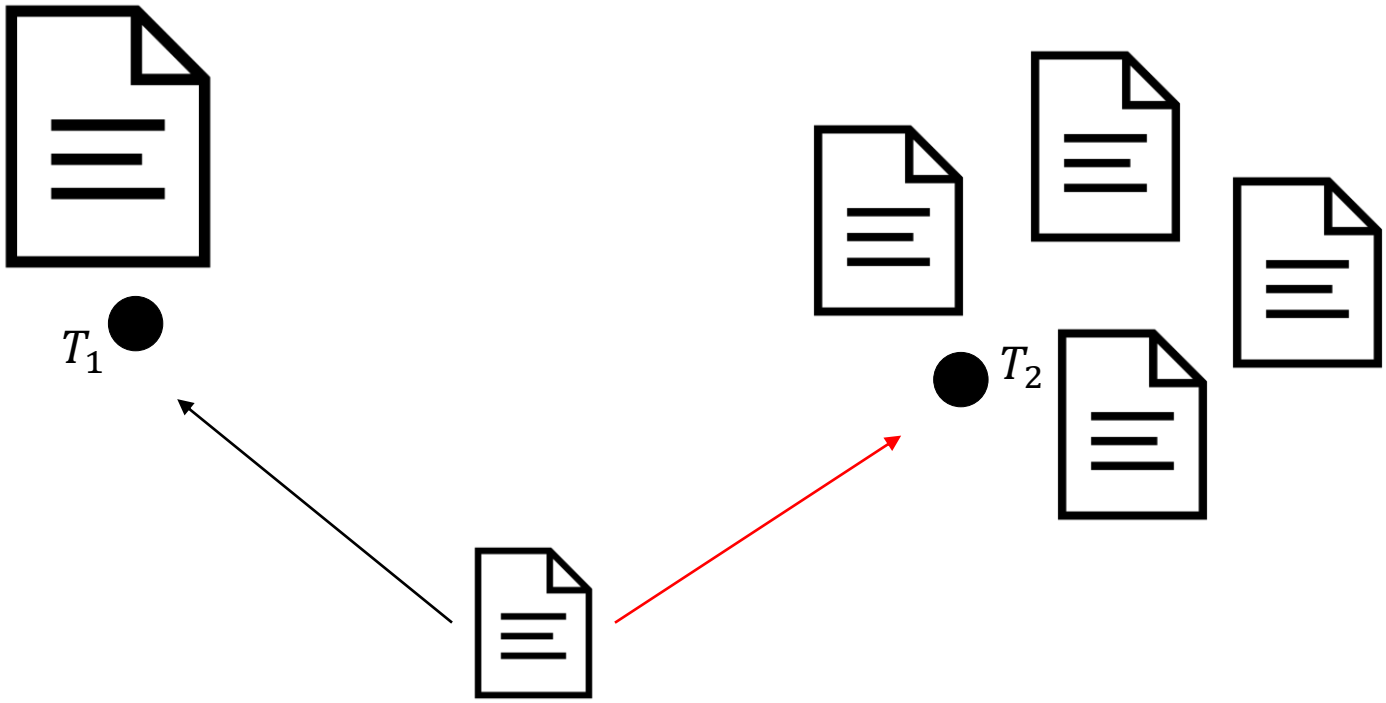


$$tf(t_1) = 9 = TRUE$$
$$tf(t_2) = 10 = TRUE$$

$$d_1 = \{1,1\}$$

Document Clustering/Classification: Term Frequency

- **What would be the weights?**
- **Where should the new document go to? Topic T_1 or T_2?**

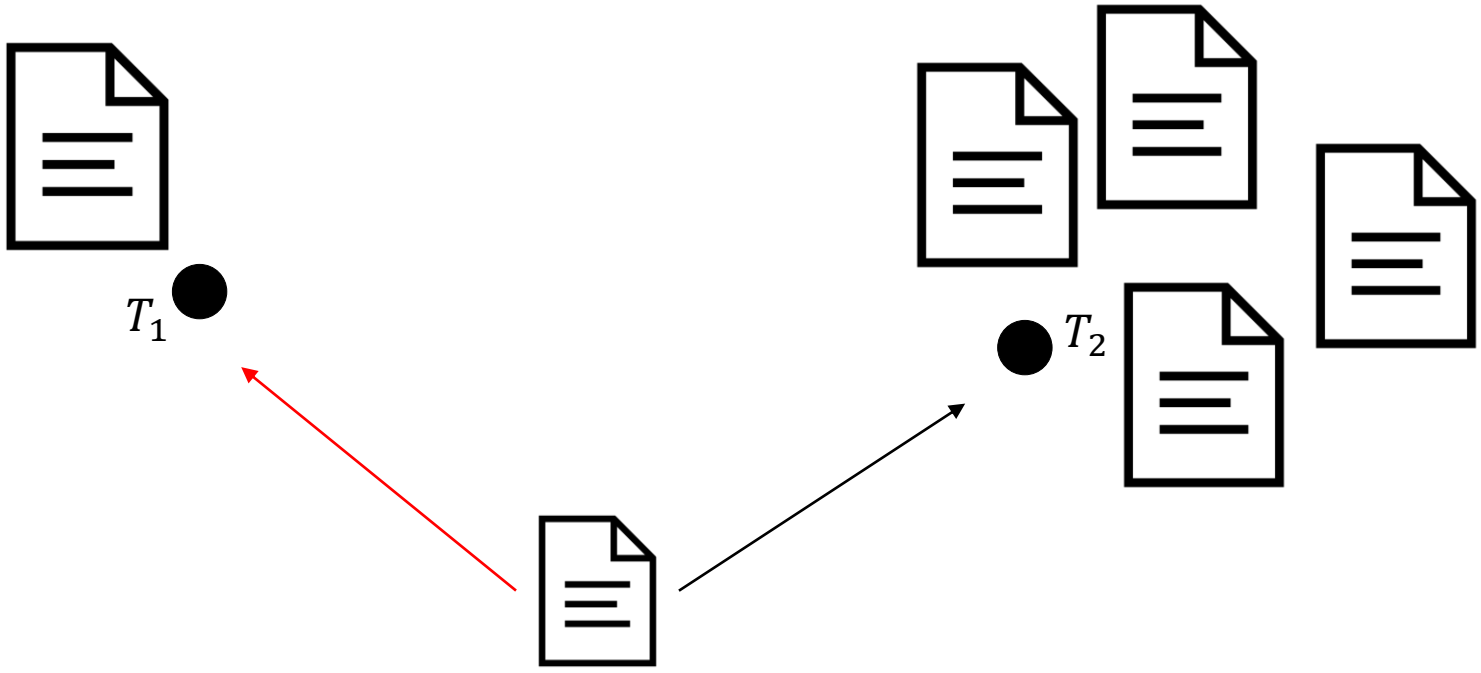


$tf(t_1) = 9$
$tf(t_2) = 10$

$d_1 = \{9,10\}$

Document Clustering/Classification: TF-IDF

- **What would be the weights?**
- **Where should the new document go to? Topic T_1 or T_2?**



$tf(t_1) = 9$
$tf(t_2) = 10$

$d_1 = \{\ldots, \ldots\}$

TF-IDF

$$\log\frac{n}{|d_{t_j}|} = -\log\frac{|d_{t_j}|}{n}$$

- **TF-IDF as more advanced weighting scheme**
- $n = \boldsymbol{number\ of\ documents\ in\ the\ corpus}$
- $|d_{t_j}|$**= # of docs that contain term** t_j

$$w_{i,j} = TFIDF(d_i, t_j) = tf(d_i, t_j) * \log\frac{n}{|d_{t_j}|}$$

$\log(1) = 0$
$\log(1{,}000{,}000{,}000) = 9$

Terms/Features 1…m

Documents/ Instances 1…n

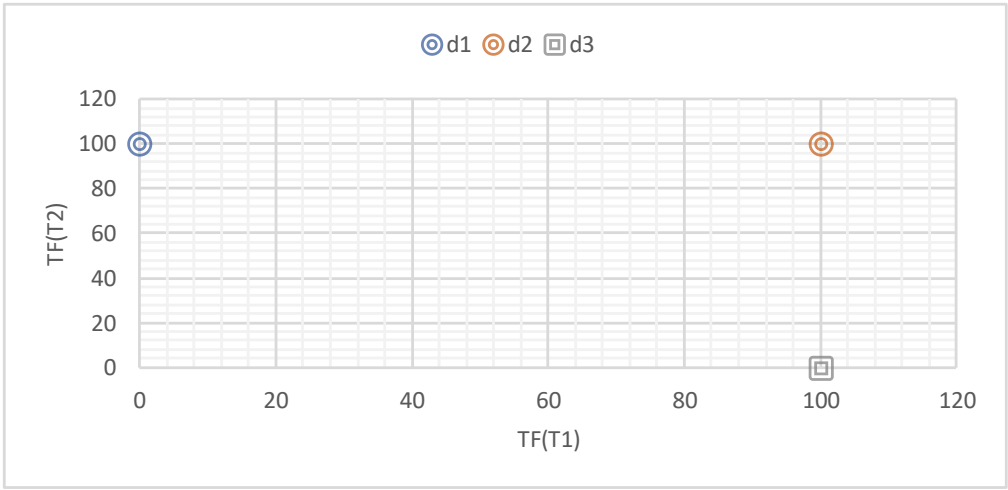
$$\begin{matrix} w_{1,1} & \cdots & w_{1,m} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \cdots & w_{n,m} \end{matrix}$$

Example 1

$$TFIDF(d_i, t_j) = tf(d_i, t_j) * \log \frac{n}{|d_{t_j}|}$$

- **Which documents are more similar, i.e. closer in the vector space? d1;d2 or d2;d3?**

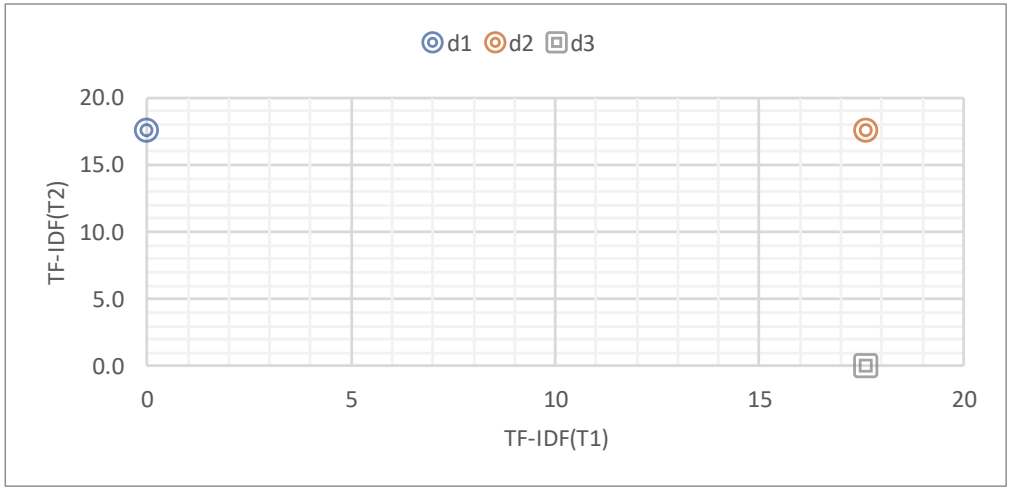
		t1	t2
TF	d1	0	100
	d2	100	100
	d3	100	0
TF-IDF	d1	0	17.6
	d2	17.6	17.6
	d3	17.6	0



$$TFIDF(d_1, t_1) = 0 * \log \frac{3}{2} = 0$$

$$TFIDF(d_2, t_1) = 100 * \log \frac{3}{2} = 17.6$$

$$TFIDF(d_3, t_1) = 100 * \log \frac{3}{2} = 17.6$$



$$TFIDF(d_i, t_j) = tf(d_i, t_j) * \log \frac{n}{|d_{t_j}|}$$

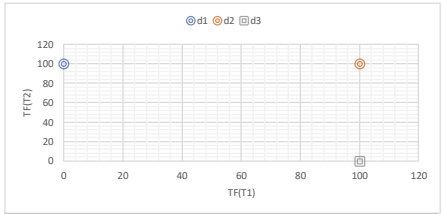
Example 2

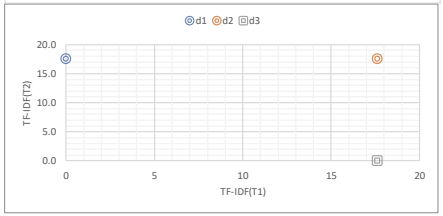
Slightly changed the distribution of term frequencies

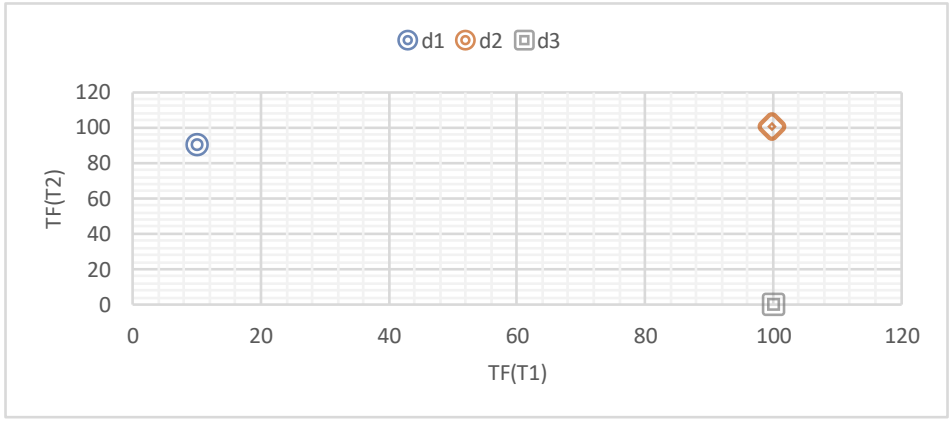
- **Which documents are more similar, i.e. closer in the vector space? d1;d2 or d2;d3?**

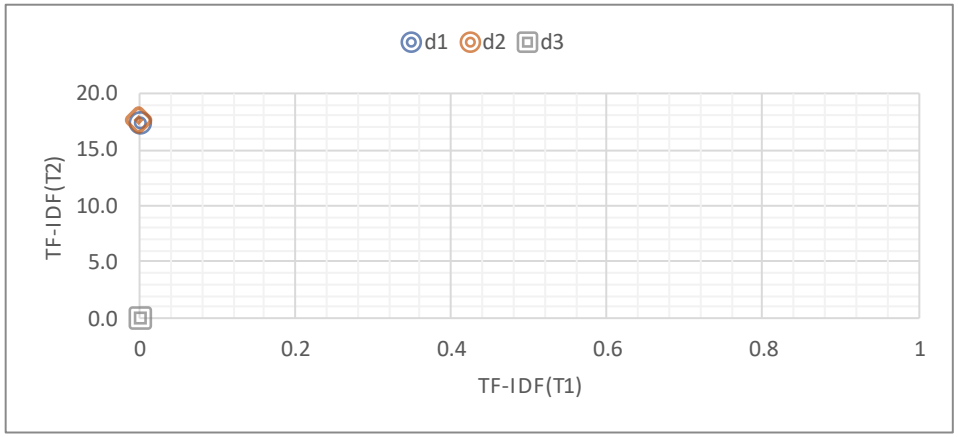
		t1	t2
TF	d1	1	99
	d2	100	100
	d3	100	0
TF-IDF	d1	0	17.4
	d2	0	17.6
	d3	0	0

		t1	t2
TF	d1	0	100
	d2	100	100
	d3	100	0
TF-IDF	d1	0	17.6
	d2	17.6	17.6
	d3	17.6	0





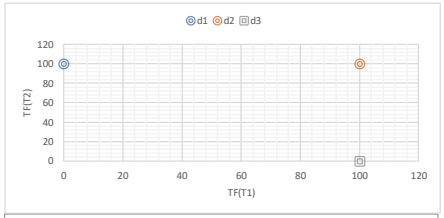


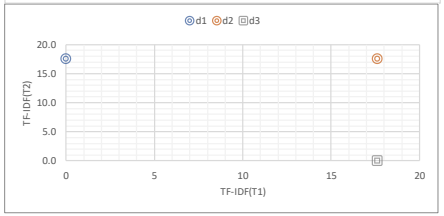


Example 3

One new document (d4) added to data from first example

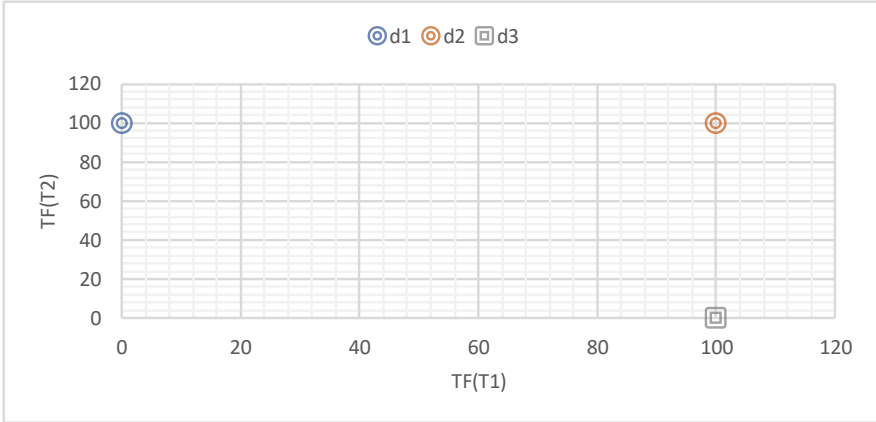
		t1	t2
TF	d1	0	100
	d2	100	100
	d3	100	0
TF-IDF	d1	0	17.6
	d2	17.6	17.6
	d3	17.6	0

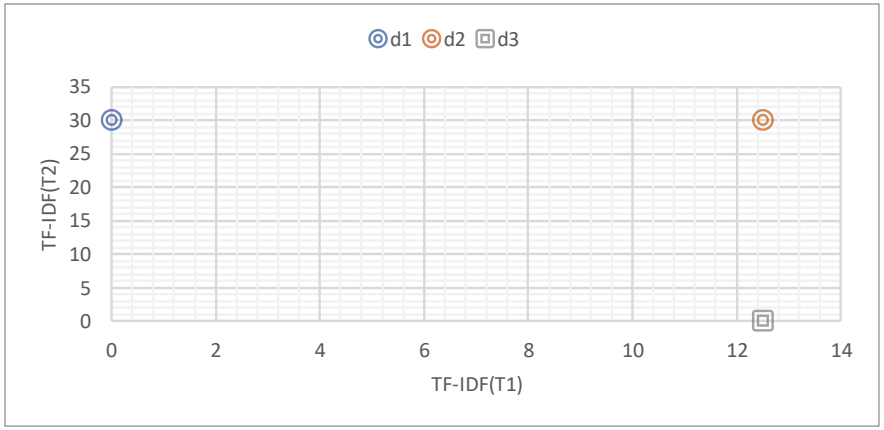




- **Which documents are more similar, i.e. closer in the vector space? d1;d2 or d2;d3?**

		t1	t2
TF	d1	0	100
	d2	100	100
	d3	100	0
	d4	1	0
TF-IDF	d1	0	30
	d2	12	30
	d3	12	0
	d4	0.12	0





Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Thank you

Beyond TF-IDF

- **https://www.kdnuggets.com/2018/08/topic-modeling-lsa-plsa-lda-lda2vec.html**
- **https://www.kdnuggets.com/2018/08/word-vectors-nlp-glove.html**

Example: Document Similarity

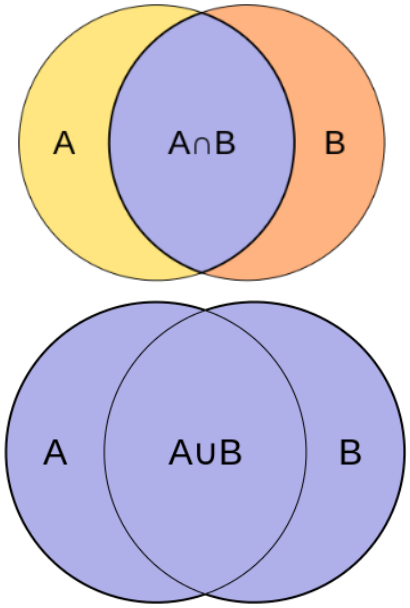
Jaccard Index

1. **Absolute overlap (absolute number of shared features)**
2. **Jaccard index (relative number of shared features)**

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

- Jaccard Distance: 1- Jaccard Index

3. **Vector Space and TF-IDF Weighting (next slide)**

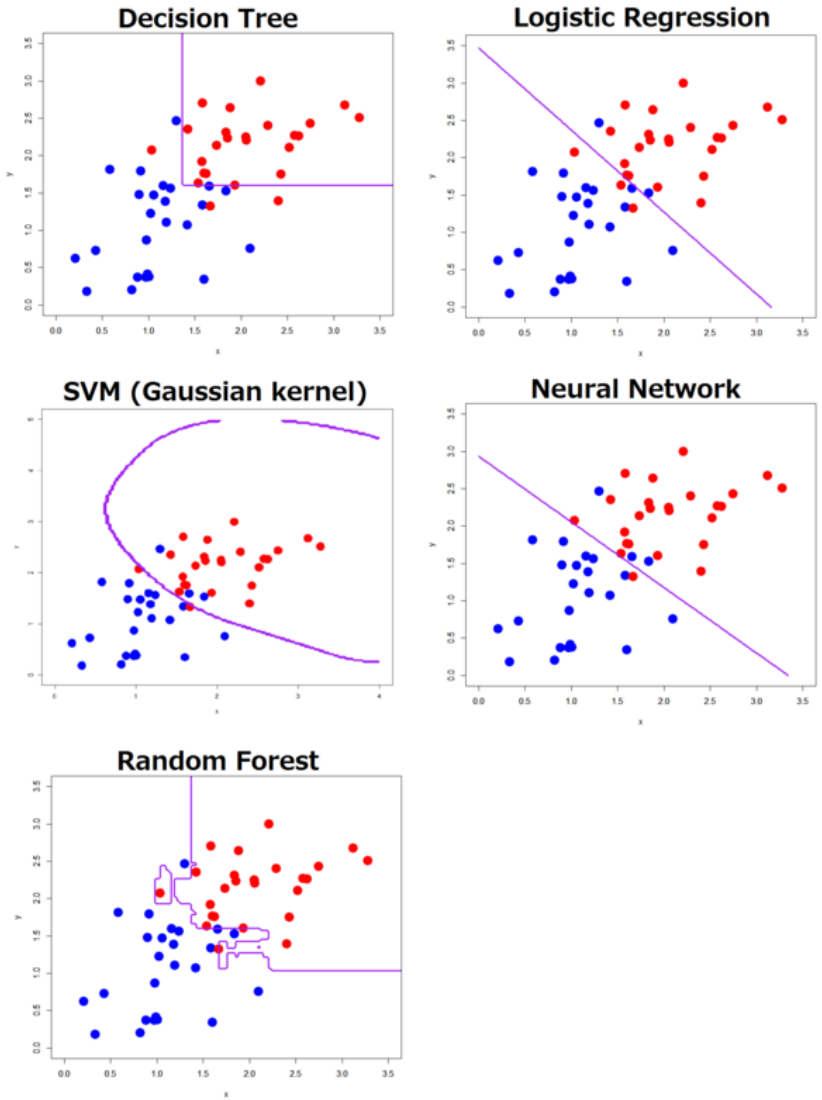


https://en.wikipedia.org/wiki/Jaccard_index

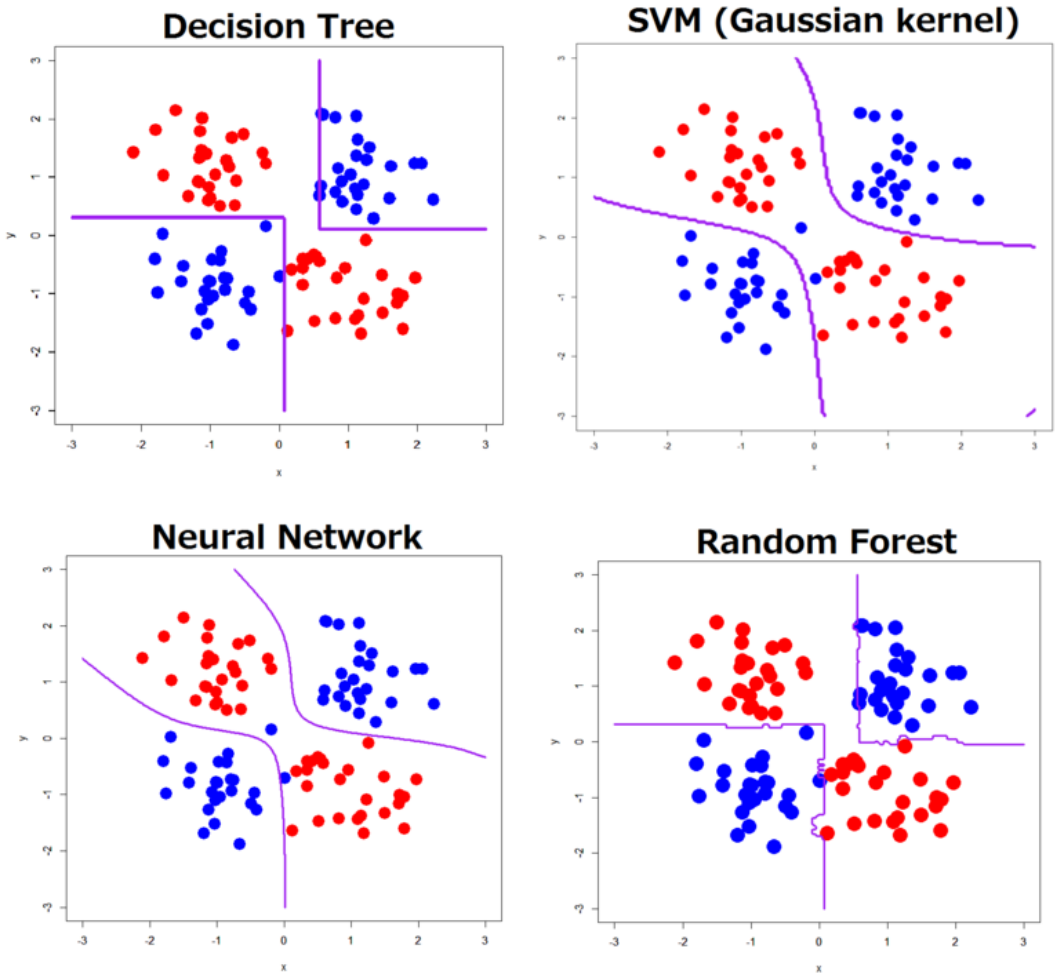
Neural Networks/Perceptron Vs. …

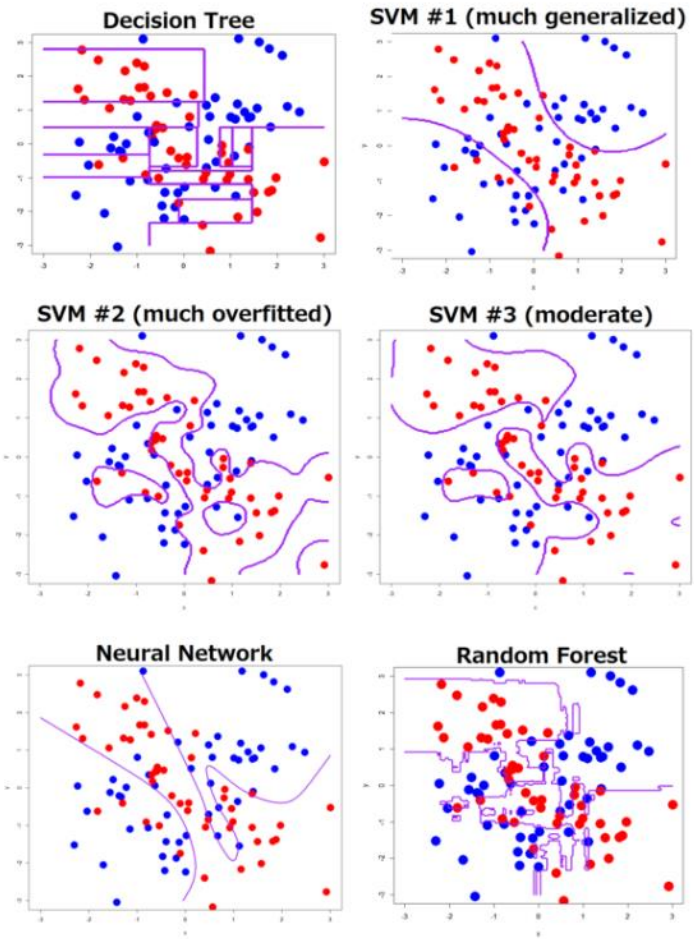
http://tjo-en.hatenablog.com/entry/2014/01/06/234155

Binary Classification (Linearly Separable)

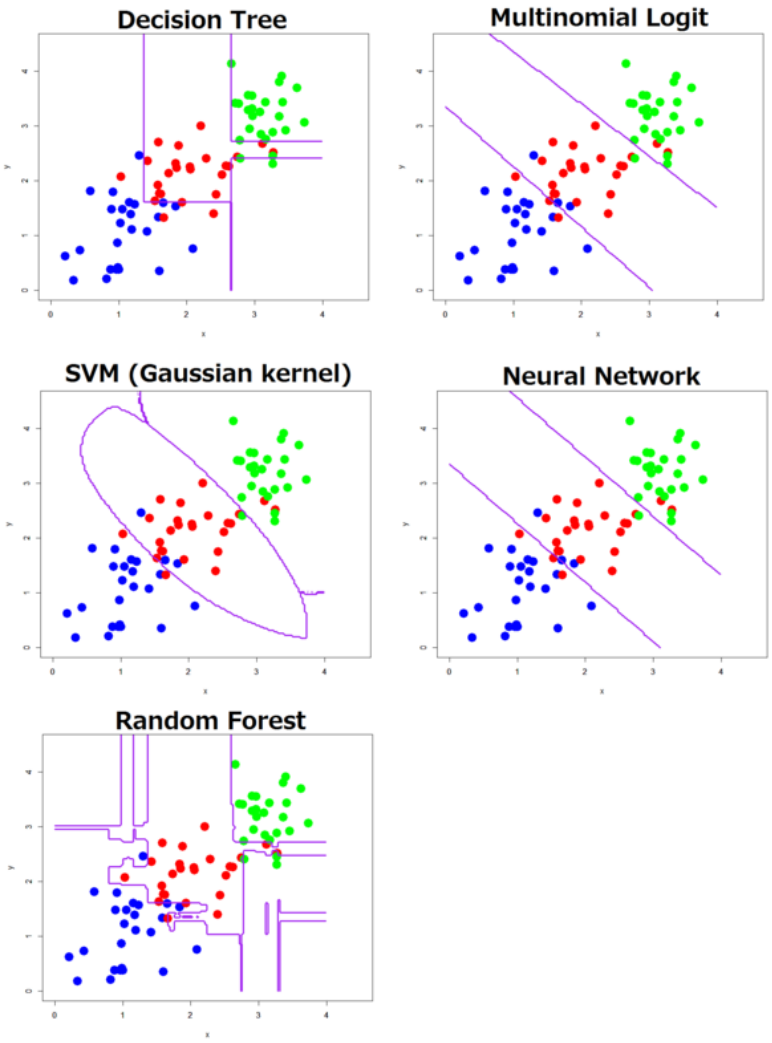


Binary Classification (Linearly Inseparable)

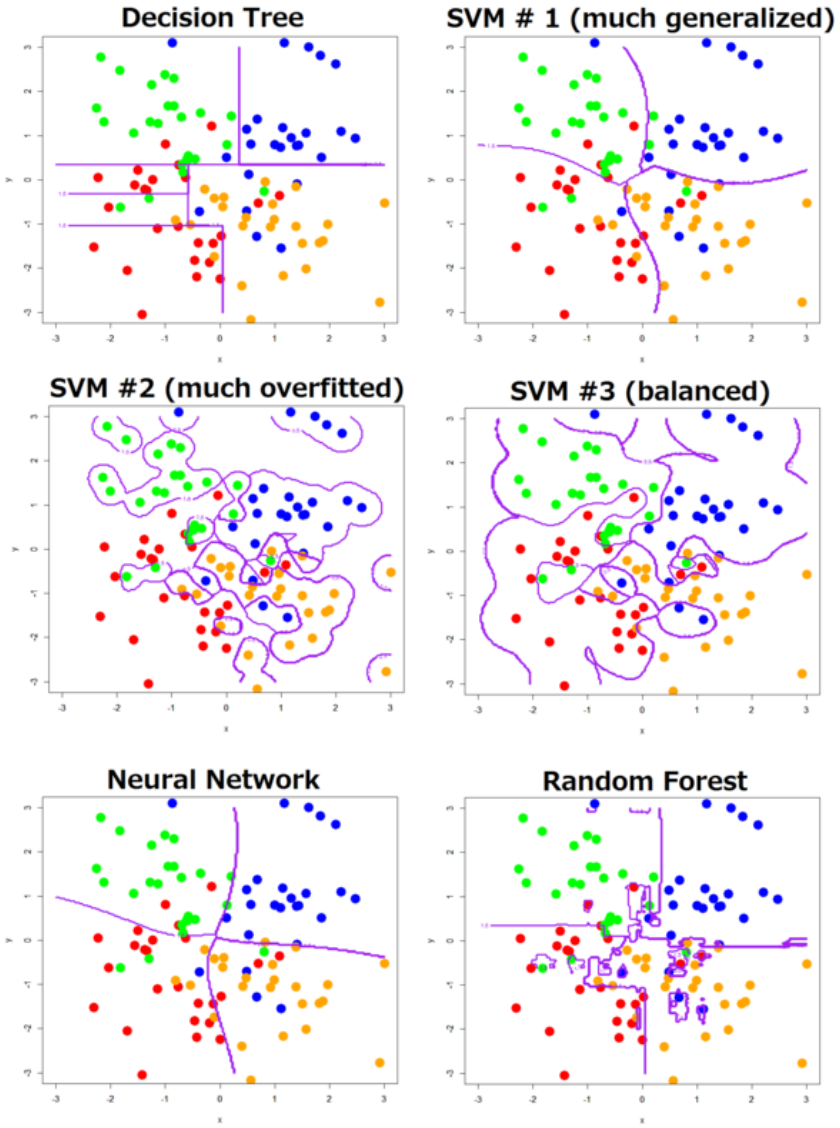




3-Class Classification (Linearly separable)



4-Class Classification



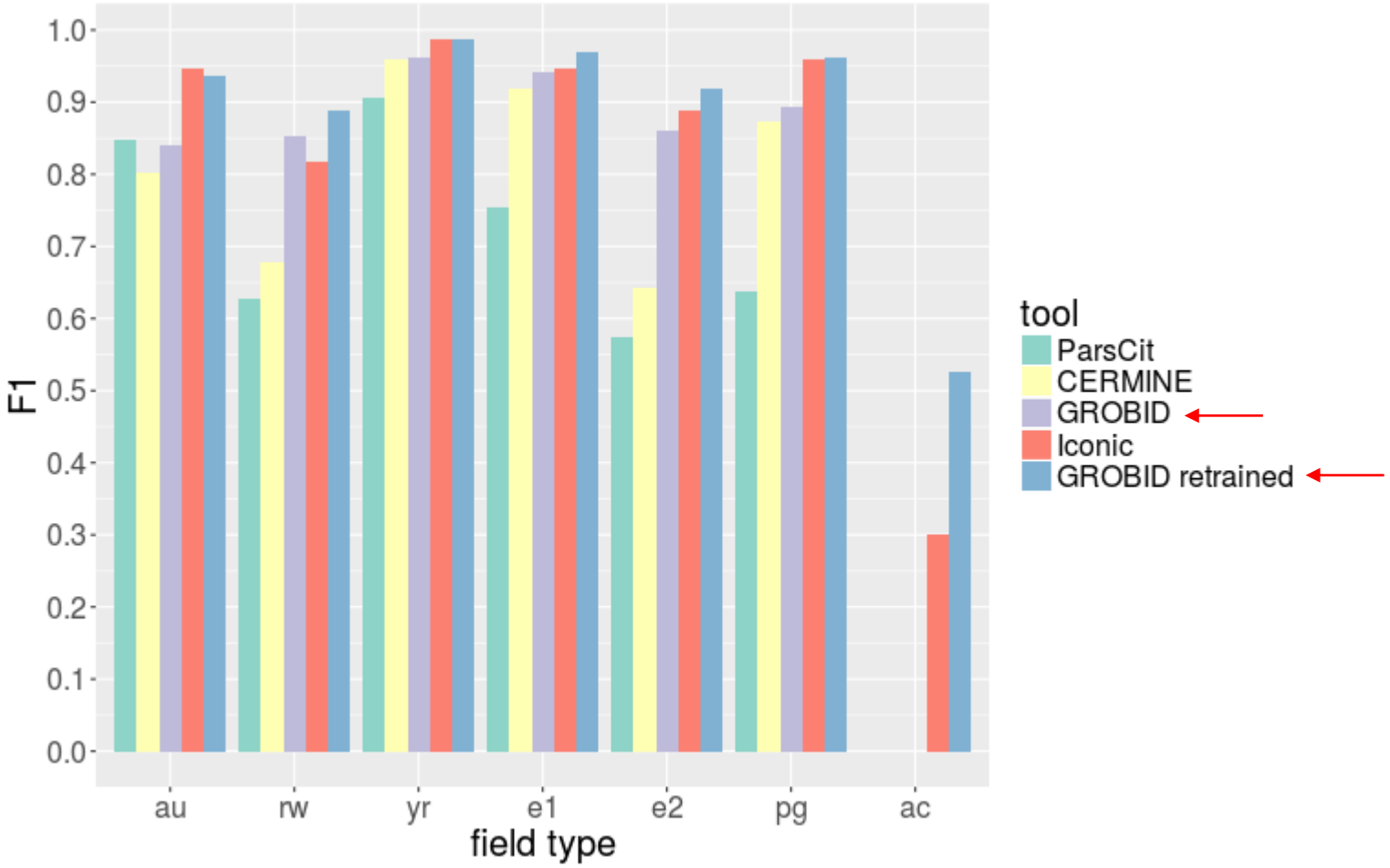
Updates

The Effect of Retraining

Juliane Stiller, Marlies Olensky, and Vivien Petras. 2014. A Framework for the Evaluation of Automatic Metadata Enrichments. In Sissi Closs, Rudi Studer, Emmanouel Garoufallou, & Miguel-Angel Sicilia, eds. *Metadata and Semantics Research: 8th Research Conference, MTSR 2014, Karlsruhe, Germany, November 27-29, 2014. Proceedings.* Cham: Springer International Publishing, 238–249. DOI:https://doi.org/10.1007/978-3-319-13674-5_23

Kazunari Sugiyama and Min-Yen Kan. 2010. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th ACM/IEEE Annual Joint Conference on Digital Libraries (JCDL).* ACM, 29–38.

Roberto Torres, Sean M. McNee, Mara Abel, Joseph A. Konstan, and John Riedl. 2004. Enhancing digital libraries with TechLens+. In *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries.* ACM New



Update (II)

Statistics

https://pbs.twimg.com/media/DDaxHcTW0AA8jVO.jpg