

Artificial Neural Networks

Course-1

Gavrilit Dragos

rev 1.0

π

AGENDA FOR TODAY

- › Administrative
- › What is AI ?
- › Basic linear algebra notions
- › History of Neural Networks

Administrative

Administrative

- › Course examination : maximum of 30 points
- › Lab examination: 60 maximum points (project grade).
- › We will use a Gauss-like system for the final grade

- › Minim requirements for passing:
 - 10 points on course examination
 - 20 points for lab participation)

Administrative

- › Technologies used:
 - Python 3.x
 - PyTorch
 - numpy
 - One native language (C/C++ or Rust)

What is AI ?

What is AI ?

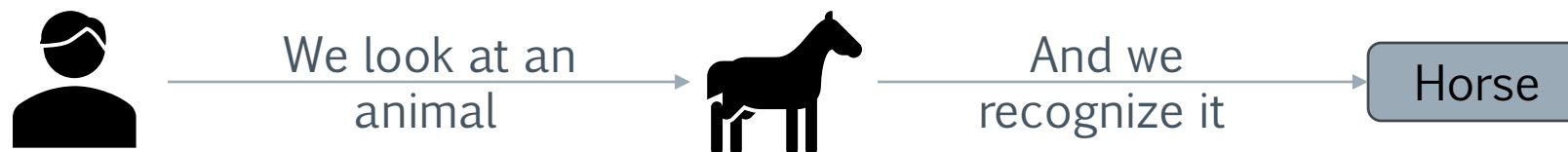
A.I.(Artificial Intelligence) refers to the simulation of human intelligence in machines.

In practice – we can consider this form of simulation an approximation on what we (human) consider intelligence.

π

What is AI ?




Let's see a simple example:



How can we translate this behavior/functionality to a computer ?

What is AI ?






Let's see a simple example:

Human behavior	Computer behavior
 → We look at an animal →  → And we recognize it → Horse	

1. First, we need the ability “to look” → this simply translate into taking a picture

What is AI ?

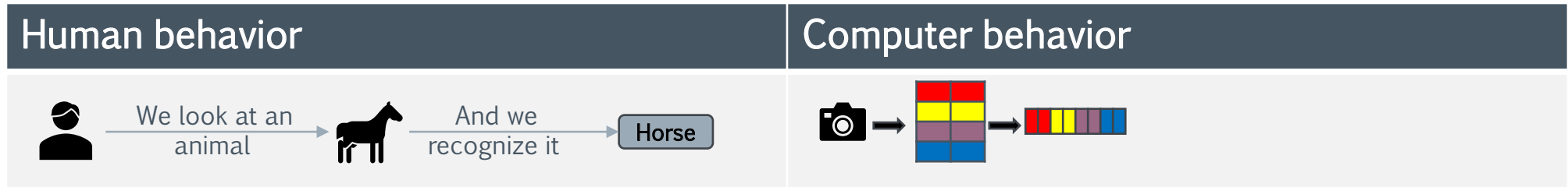
Let's see a simple example:

Human behavior	Computer behavior
 → We look at an animal →  → And we recognize it → 	 → 

1. First, we need the ability “to look” → this simply translate into taking a picture
2. That picture is transferred to a computer where it is converted into a matrix of ($w \times h$), where “ w ” is the width of the picture and “ h ” is its height. Each cell of that matrix is a color (a numerical value) that reflects the color of the pixels from the original picture.

What is AI ?

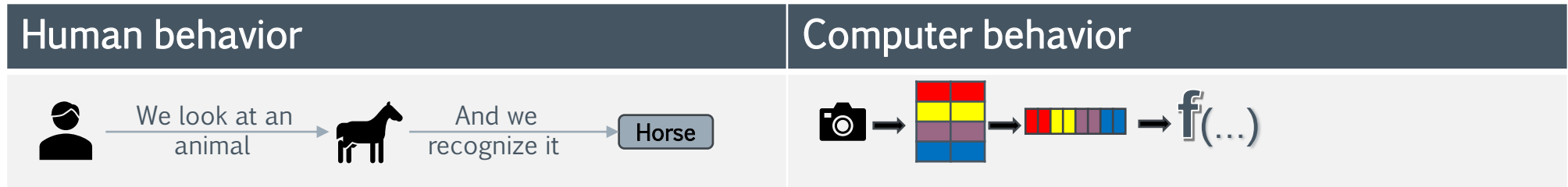
Let's see a simple example:



1. First, we need the ability “to look” → this simply translate into taking a picture
2. That picture is transferred to a computer where it is converted into a matrix of ($w \times h$), where “ w ” is the width of the picture and “ h ” is its height. Each cell of that matrix is a color (a numerical value) that reflects the color of the pixels from the original picture.
3. We convert the matrix into a vector of numbers by concatenating each row one after another

What is AI ?

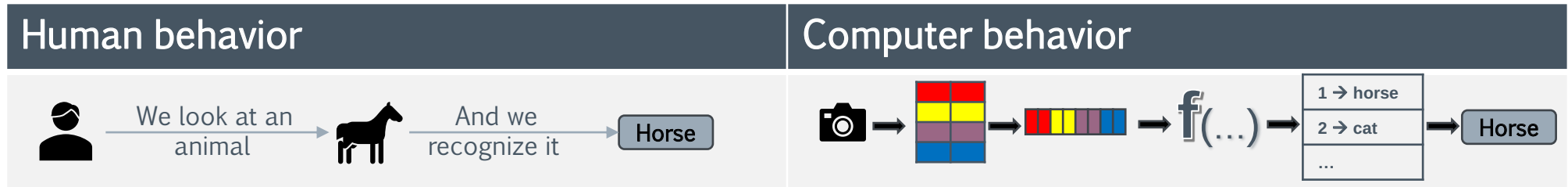
Let's see a simple example:



1. First, we need the ability “to look” → this simply translate into taking a picture
2. That picture is transferred to a computer where it is converted into a matrix of ($w \times h$), where “ w ” is the width of the picture and “ h ” is its height. Each cell of that matrix is a color (a numerical value) that reflects the color of the pixels from the original picture.
3. We convert the matrix into a vector of numbers by concatenating each row one after another
4. Now we need a function that takes “ w ” x “ h ” parameters (let's call it function f), and returns a number with a very special meaning: if the result is 1, it corresponds to the word “horse”, if the result is 2, it corresponds to the word “cat” and so on.

What is AI ?

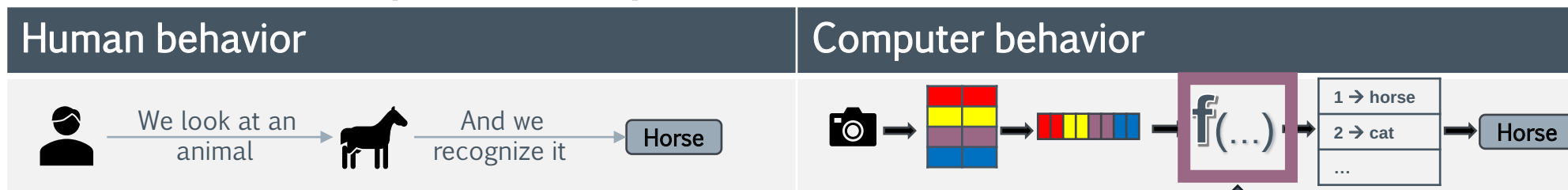
Let's see a simple example:



1. First, we need the ability “to look” → this simply translate into taking a picture
2. That picture is transferred to a computer where it is converted into a matrix of ($w \times h$), where “ w ” is the width of the picture and “ h ” is its height. Each cell of that matrix is a color (a numerical value) that reflects the color of the pixels from the original picture.
3. We convert the matrix into a vector of numbers by concatenating each row one after another
4. Now we need a function that takes “ w ” x “ h ” parameters (let's call it function f), and returns a number with a very special meaning: if the result is 1, it corresponds to the word “horse”, if the result is 2, it corresponds to the word “cat” and so on.
5. Finally, we use a map with associations ($1 \rightarrow$ horse, $2 \rightarrow$ cat, ...) and we print the result.

What is AI ?

Let's see a simple example:



1. First, we need the ability “to look” → this simply translate into taking a picture
2. That picture is transferred to a computer where it is processed. The image is represented as a matrix where “ w ” is the width of the picture and “ h ” is its height (a numerical value) that reflects the color of the pixels.
3. We convert the matrix into a vector of numbers by concatenating each row one after another.
4. Now we need a function that takes “ w ” x “ h ” parameters (let's call it function f), and returns a number with a very special meaning: if the result is 1, it corresponds to the word “horse”, if the result is 2, it corresponds to the word “cat” and so on.
5. Finally, we use a map with associations (1 → horse, 2 → cat, ...) and we print the result.

This is where the *magic* happens.
The question is how do we find this function or if it exists ?

What is AI ?

Let's make this more generic – in the previous example, the function “***f***” was meant to return one value.

In practice, depending on what kind of task we want to achieve, “***f***” can return multiple values:

$$f: R^n \rightarrow R^m, \text{ with } n, m \in N, \text{ and } n \geq 1, m \geq 1$$

What is AI ?

So ... the question here is \rightarrow *Is there a function f that can satisfy our requirements ?*

The answer is that **we should not search for an exact function, but one that approximates the results as much as possible !**

The function that we find (meaning its form and its parameters) is called a **model** !

What is AI ?

Let's consider that for a specific problem, we manage to find a function ***f*** that looks like the following:

$$f(x, y, z) = 5 \times x + 9 \times y - 10 \times z,$$

For this example, we can say that the model is **[5, 9 and -10]**

What is AI ?

So ... the new question is how can we approximate a function/model that has to obtain a specific value ?

Or even more, assuming we have such a function/model, how can we measure its effectiveness ?

What is AI ?

To answer both these questions – we will need a dataset.

A data set (for classification) is a set of entries where we know their output. For example, assuming we are trying to find the function “***add***” for two numbers, the data set will look like this:

Input for the function			
Entry	x_1	x_2	Output (x_1+x_2)
#1	5	20	25
#2	2	7	9
#3	-11	100	89
#4	30	30	60
#5	0	0	0
...
#n	12	11	23

What is AI ?

Having access to a database allows us to:


1. Use several techniques to approximate the output as best as we can
2. Validate that the output is the one that we expect.

What is AI ?

Having access to a database allows us to:

1. Use several techniques to approximate the output as best as we can

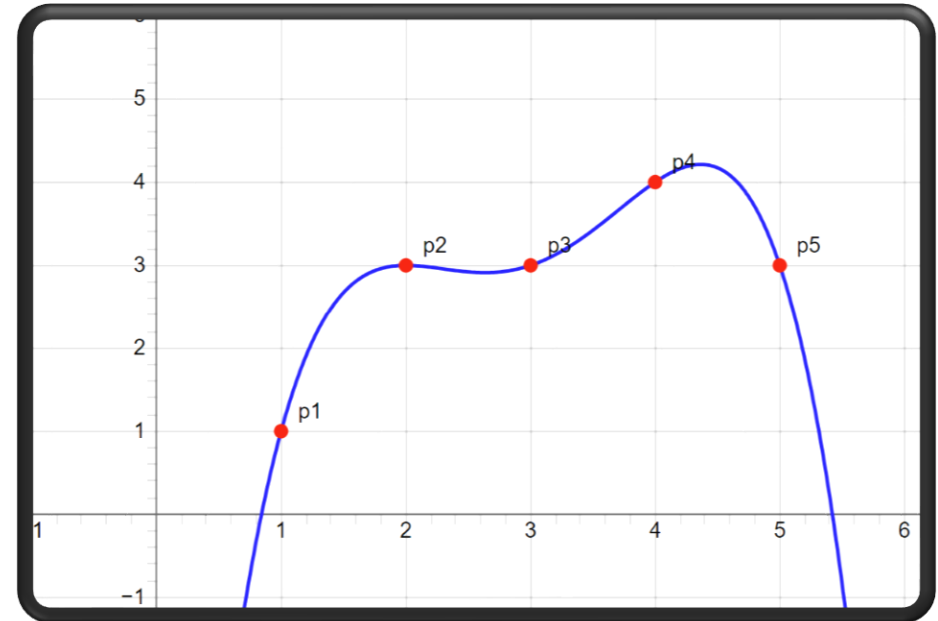
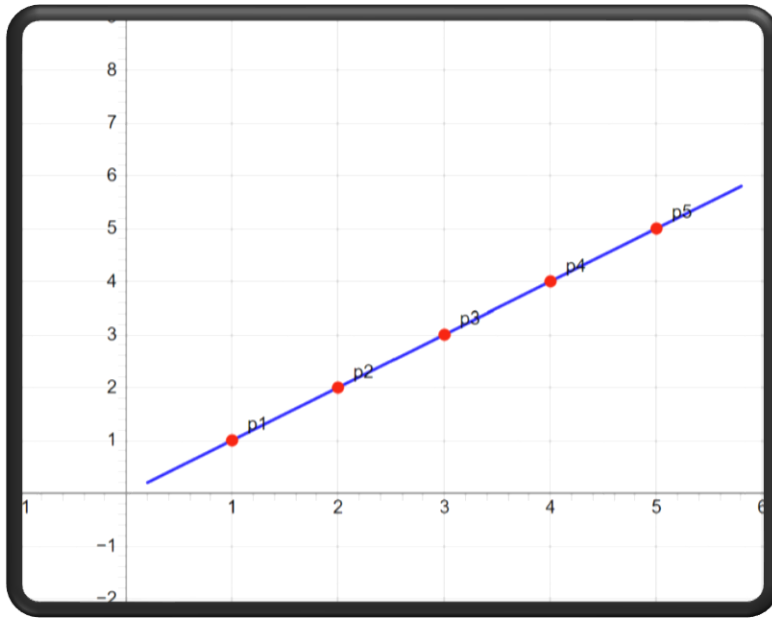
2. Validate that the output is the one that we expect.



One simple example is the **Lagrange Interpolation** technique (that even if initially design to work with 2-dimensional points) is a good start in understanding how A.I. works.

What is AI ?

Lagrange interpolation – given a set of points, identify a function “ f ” that approximates “ Y ” given an “ X ” $\rightarrow f(x) = y$



What is AI ?

Having access to a database allows us to:

1. Use several techniques to approximate the output as best as we can
2. Validate that the output is the one that we expect.



The main question here is *how can we measure* this ?

What is AI ?

How to validate A.I. ?

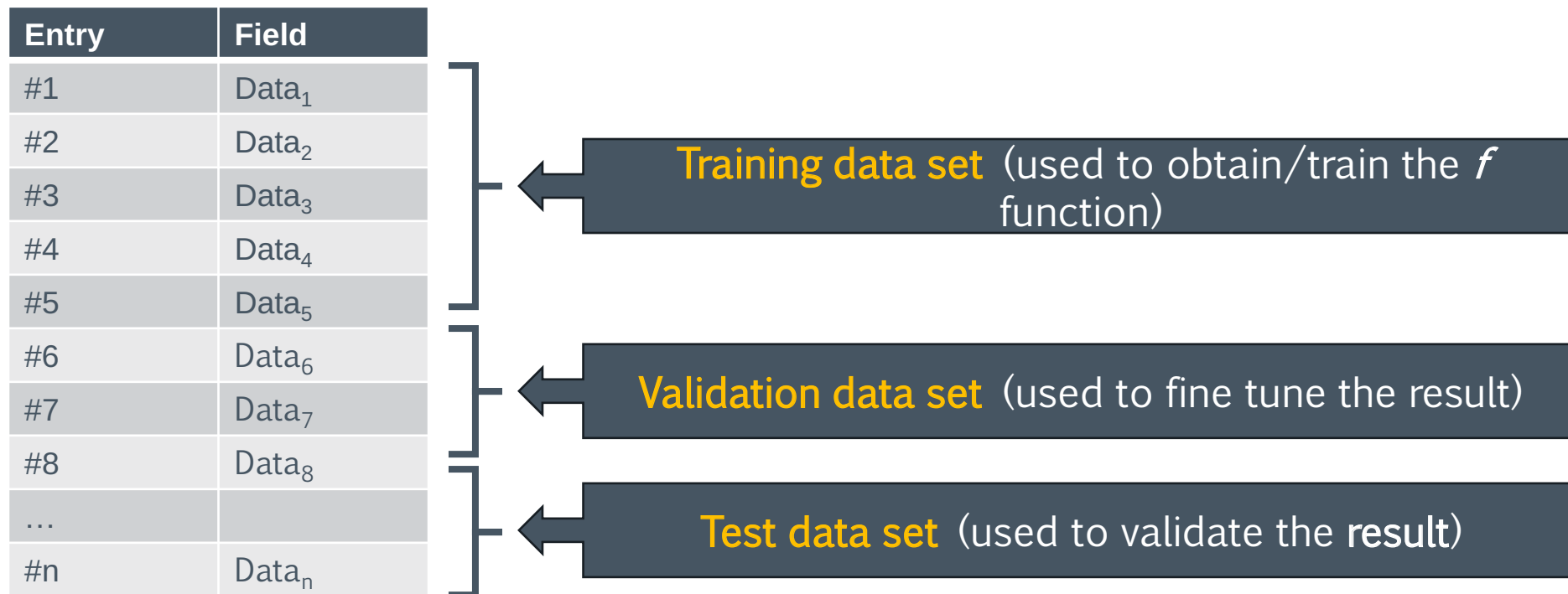
There are several options that can be used to test this:

1. The database has to be splitted into several parts (some to be used to identify the function “ f “, others for validation of that function)
2. We will need some metrics (to validate how good the resulted function is).

What is AI ?

How to validate A.I. ?

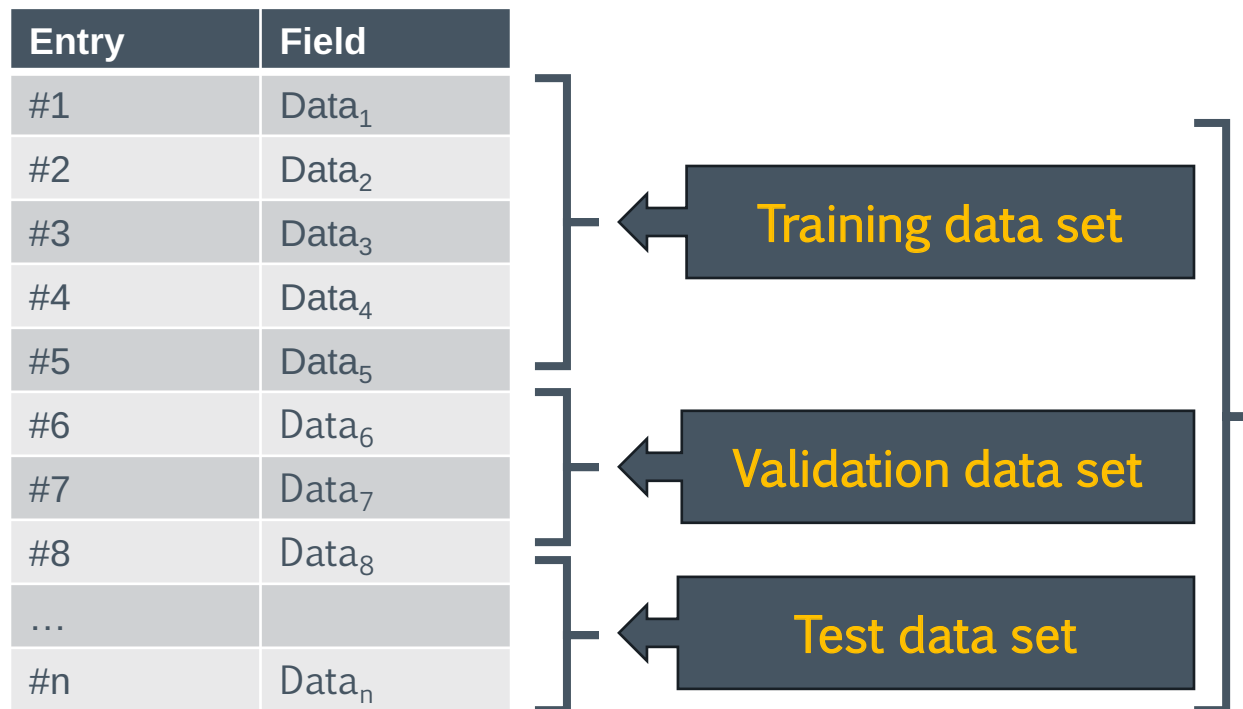
First – lets see how we can split a database:



What is AI ?

How to validate A.I. ?

First – lets see how we can split a database:



It's important to **select a proper balance** between these data sets. For example, if the training set is too small while the test set is too large, the results will be inconclusive as the algorithm will not have enough data to learn from. Similarly, if no test/validation data set is available (or too small), the results may not be validated properly, or an **overfitting** state can be achieved.

What is AI ?

Overfitting

Overfitting refers to a modeling error that occurs when a machine learning algorithm captures noise in the training data rather than the underlying data distribution. This results in a model that performs well on the training data but poorly on new data, because it has essentially "memorized" the training data rather than generalizing from it.

What is AI ?

Overfitting

Let's see an example (we will consider a data set of 4 entries where the first 3 entries are used for training, and the last one for validation).

Dataset	x_1	$F(x_1)$
#1 (training)	1	2
#2 (training)	2	4
#3 (training)	3	6
#4 (test/validation)	4	8

Case 1

$$f(x) = x \times 2$$

Generic solution
 $f(4) = 8$

Case 2

$$f(x) = \begin{cases} 2, & x = 1 \\ 4, & x = 2 \\ 6, & x = 3 \\ 0, & x \notin \{1, 2, 3\} \end{cases}$$

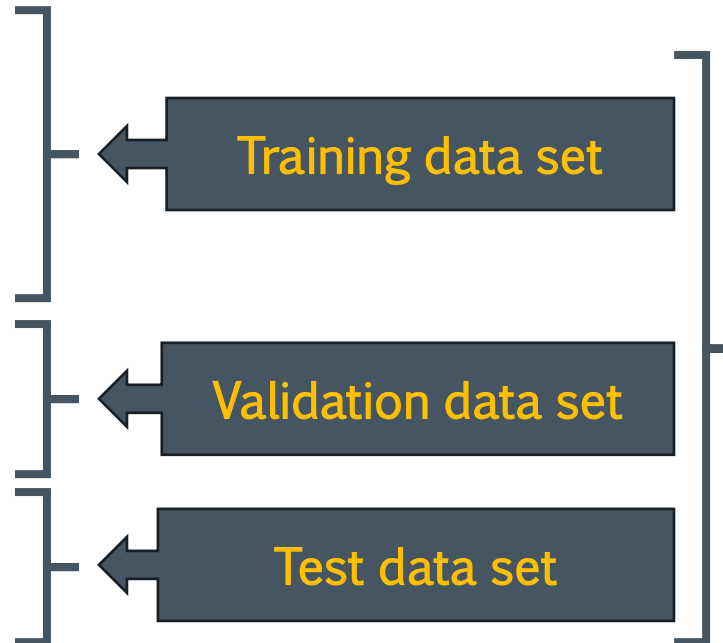
Overfitting
 $f(4) = 0$

What is AI ?

How to validate A.I. ?

First – lets see how we can split a database:

Entry	Field
#1	Data ₁
#2	Data ₂
#3	Data ₃
#4	Data ₄
#5	Data ₅
#6	Data ₆
#7	Data ₇
#8	Data ₈
...	
#n	Data _n



The **validation data set** is **OPTIONAL**

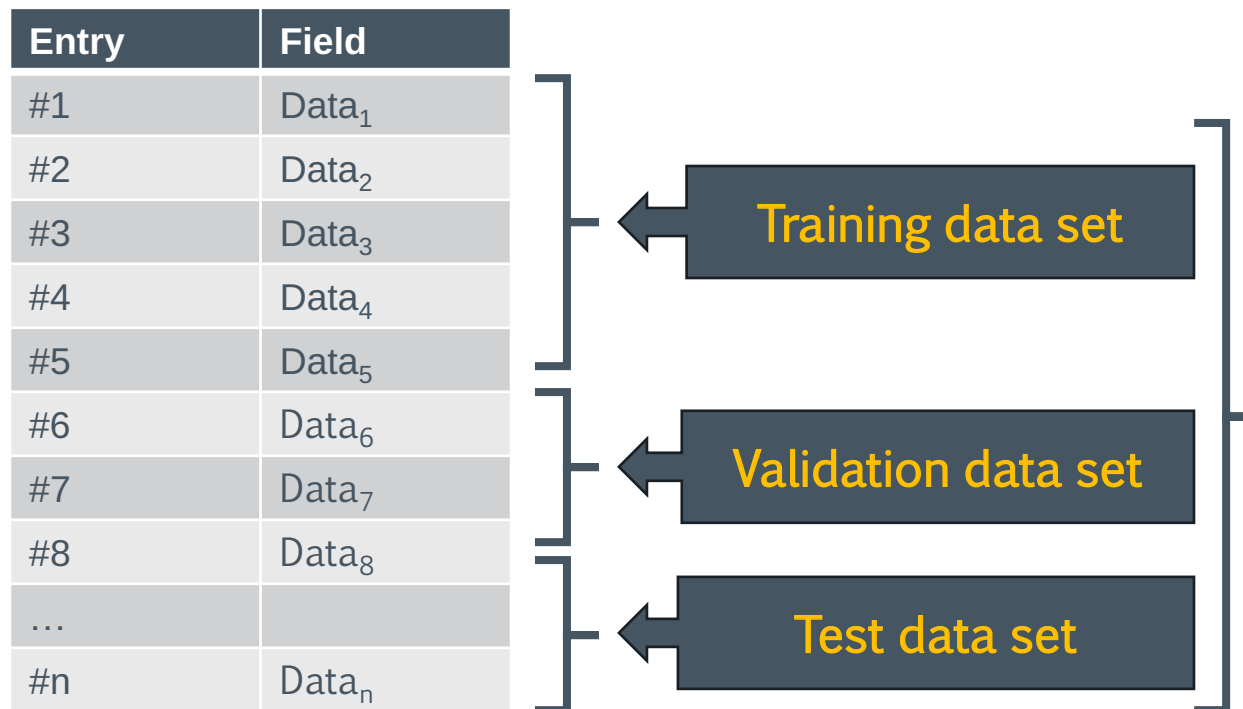
Some percentages commonly used:

- **60/20/20** (60% training, 20% validation, 20% testing)
- **70/15/15** (70% training, 15% validation, 15% testing)
- **80/10/10** (80% training, 10% validation, 10% testing)

What is AI ?

How to validate A.I. ?

First – lets see how we can split a database:



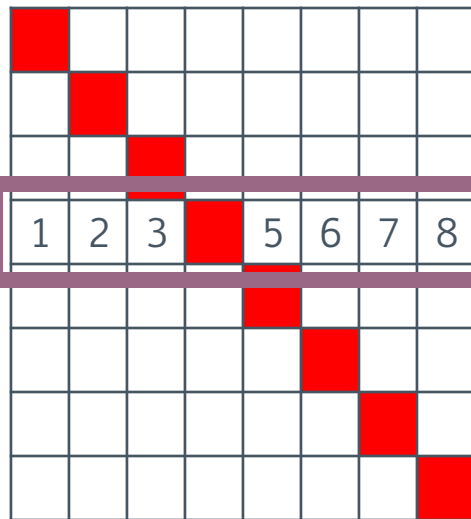
The **validation** and **testing** data set don't have to be equals.

- If you perform hyper-parameter tuning → it is recommended to use a larger validation data set
- If you want the result to be more generic, and have a better check and a more reliable result → it is recommended to use a larger test data set.

What is AI ?

How to validate A.I. ?

Another solution on how to split a database is to use a technique called k-fold cross-validation. The idea is to split the entire database into “k” partition, and then use (k-1) partitions for training and the partition that remains for testing. The validation is performed for “k” times (each time using a different partition for testing).



For example, in this case we use partition 1,2,3 5,6,7,8 to train and partition 4 for testing (in a **8-fold cross validation**)

What is AI ?

How to validate A.I. ?

Before we discuss metrics to evaluate, lets first discuss about type of artificial intelligence algorithms:

- › **Supervised learning** - the model is trained on a labeled dataset, meaning the algorithm is provided with input-output pairs.
- › **Unsupervised learning** - works with unlabeled data. The goal is to find patterns or structures in the data. One such example are clustering algorithms
- › **Other variations:**
 - Reinforcement learning
 - Semi-supervised learning
 - ...

What is AI ?

How to validate A.I. ?



When talking about ***supervised learning*** a machine learning algorithm can be classify (taking into consideration the output) in:

- › **Single class (binary models)** - Classify instances into one of two classes. For example , in our example that tries to identify a picture and find a horse, the output of a function can be 1 (**TRUE**) or 0 (**FALSE**).
- › **Multi class** - Classify instances into one of more than two classes. For the same example, the output of the function could be 1 (for a Horse), 2 for a Monkey, 3 for a Cat,

What is AI ?

How to validate A.I. ?

To better explain how several metrics work, lets consider a supervised single class machine learning that attempts to identify pictures with cats from a data base.

		Response is that the picture is with a cat.	Response is that the picture is NOT with a cat.
	Picture with a cat	TRUE POSITIVE	FALSE NEGATIVE
	Picture with an eagle	FALSE POSITIVE	TRUE NEGATIVE

What is AI ?

How to validate A.I. ?

Let's assume that we have the following data base:

Index	Actual label / Desired output	Result of the model
#1	CAT	CAT
#2	CAT	Not A Cat
#3	Not A Cat	CAT
#4	CAT	CAT
#5	Not A Cat	Not A Cat
#6	Not A Cat	Not A Cat
#7	Not A Cat	CAT
#8	CAT	CAT
#9	CAT	CAT

True Positives (TP) = 4
(entries #1, #4, #8, #9)

What is AI ?

How to validate A.I. ?

Let's assume that we have the following data base:

Index	Actual label / Desired output	Result of the model
#1	CAT	CAT
#2	CAT	Not A Cat
#3	Not A Cat	CAT
#4	CAT	CAT
#5	Not A Cat	Not A Cat
#6	Not A Cat	Not A Cat
#7	Not A Cat	CAT
#8	CAT	CAT
#9	CAT	CAT

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

What is AI ?

How to validate A.I. ?

Let's assume that we have the following data base:

Index	Actual label / Desired output	Result of the model
#1	CAT	CAT
#2	CAT	Not A Cat
#3	Not A Cat	CAT
#4	CAT	CAT
#5	Not A Cat	Not A Cat
#6	Not A Cat	Not A Cat
#7	Not A Cat	CAT
#8	CAT	CAT
#9	CAT	CAT

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

What is AI ?

How to validate A.I. ?

Let's assume that we have the following data base:

Index	Actual label / Desired output	Result of the model
#1	CAT	CAT
#2	CAT	Not A Cat
#3	Not A Cat	CAT
#4	CAT	CAT
#5	Not A Cat	Not A Cat
#6	Not A Cat	Not A Cat
#7	Not A Cat	CAT
#8	CAT	CAT
#9	CAT	CAT

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

Accuracy (Acc)

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} = \frac{4 + 2}{4 + 2 + 1 + 2} = \frac{6}{9} = 66\%$$

accuracy is a measure of the correctness of a model's predictions , in other words how many samples were correctly classified from the test set. It is good for a balanced data set.

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN} = \frac{2}{2 + 2} = \frac{2}{4} = 50\%$$

proportion of negative instances that are incorrectly
classified as positive

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

False Negative Rate (FNR)

$$FNR = \frac{FN}{FN + TP} = \frac{1}{1 + 4} = \frac{1}{5} = 20\%$$

proportion of positive instances that are incorrectly
classified as negative

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

Precision

$$Precision = \frac{TP}{TP + FP} = \frac{4}{4 + 2} = \frac{4}{6} = 66\%$$

evaluate the correctness of positive predictions (it is in particular valuable for models where TP is relevant → for example a classifier that tries to determine if a disease is present or not).

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

Recall or Sensitivity (Se)

$$Se = \frac{TP}{TP + FN} = \frac{4}{4 + 1} = \frac{4}{5} = 80\%$$

Shows how many positive instances the model might be missing). Relevant for scenarios where failing to identify a positive instance has significant consequences. Also known as **True Positive Rate**.

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

Specificity (Sp)

$$Sp = \frac{TN}{TN + FP} = \frac{2}{2 + 2} = \frac{2}{4} = 50\%$$

Evaluates how well the model identifies negative instances and avoids false alarms. Also known as True Negative Rate.

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

With this values computed we can evaluate several metrics:

F1 Score

$$\begin{aligned} F1\ Score &= 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{66\% \times 80\%}{66\% + 80\%} \\ &= \frac{1.056}{1.46} = 72\% \end{aligned}$$

Useful when both false positives and false negatives are crucial and for cases where data set is unbalanced (the number of entries of one class is far bigger than the entries of another class).

True Positives (TP) = 4
(entries #1, #4, #8, #9)

True Negatives (TN) = 2
(entries #5, #6)

False Negatives (FN) = 1
(entry #2)

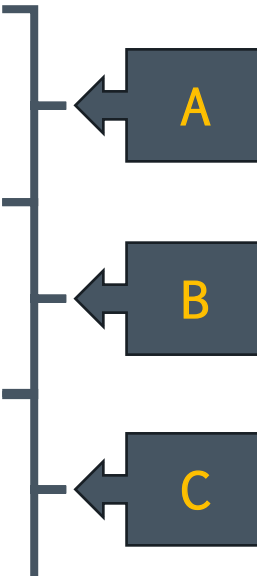
False Positives (FP) = 2
(entries #3, #7)

What is AI ?

How to validate A.I. ?

Now, lets see how a model can be validated using a 3-fold cross validation and the FPR metric. To simplify the problem lets consider that the database is formed out of 3 partition A, B and C (all equal in size).

Index	Values
#1	$V_{(1.1)} \dots V_{(1,n)}$
...	
#k	$V_{(k.1)} \dots V_{(k,n)}$
#k+1	$V_{(k+1.1)} \dots V_{(k+1,n)}$
...	
#2k	$V_{(2k.1)} \dots V_{(2k,n)}$
#2k+1	$V_{(2k+1.1)} \dots V_{(2k+1,n)}$
...	
#3k	$V_{(3k.1)} \dots V_{(3k,n)}$



Train	Test	FPR
A+B	C	5%
A+C	B	4%
B+C	A	3%

This means that the average value for FPR is $(3\% + 4\% + 5\%) / 3 = 4\%$. This value represents how well the model behaves on this data set.

Basic linear algebra notions

Basic notions

Before we start, lets discuss some basic notions used in linear algebra that we will further use in our course:

- Scalar
- Vector
- Matrix

And some operations:

- vector
- matrixes

Basic notions

1. **Scalar** = a single numerical value that describes a quantity (e.g. **100** or **5.6** or **-1.45**)
2. **Vector** = a list of scalar values. The number of scalar values within a vector is often referred to as ***dimension***.

example:

$$v = \begin{bmatrix} 10 \\ 20 \\ 30 \\ 40 \\ 50 \end{bmatrix} \text{ or } v = [10, 20, 30, 40, 50]$$

Basic notions

- 3. Matrix** = a bidimensional array of scalar elements (we can think that a matrix is a vector of vectors of scalars, with the condition that each vector of scalars have the same number of elements).

example:

$$m = \begin{bmatrix} 1 & 2 & 3 \\ 100 & 200 & 300 \end{bmatrix}, m \text{ is a } 2 \times 3 \text{ matrix}$$

Basic operations (vectors)

1. Scalar multiplication

For $v = [v_1, v_2, \dots, v_n]$ a vector, and “s” a scalar, we define:

$$v \times s = \begin{bmatrix} v_1 \times s \\ v_2 \times s \\ \vdots \\ v_n \times s \end{bmatrix}$$

Example:

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}, s = 3, \text{ then } v \times s = \begin{bmatrix} 1 \times 3 \\ 2 \times 3 \\ 3 \times 3 \\ 4 \times 3 \\ 5 \times 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 9 \\ 12 \\ 15 \end{bmatrix},$$

Basic operations (vectors)

2. Dot product

For $v = [v_1, v_2, \dots, v_n]$ a vector, and $u = [u_1, u_2, \dots, u_n]$ another vector, we define the dot product as follows:

$$v \cdot u = \sum_{i=1}^n (v_i \times u_i)$$

Example:

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, u = \begin{bmatrix} -2 \\ 4 \\ 0 \\ 3 \end{bmatrix}, \text{ then } v \cdot u = (1 \times -2) + (2 \times 4) + (3 \times 0) + (4 \times 3) = -2 + 8 + 0 + 12 = \mathbf{18}$$

Basic operations (vectors)

3. Addition/Subscription of two vectors

For $v = [v_1, v_2, \dots, v_n]$ a vector, and $u = [u_1, u_2, \dots, u_n]$ another vector, we define the dot product as follows:

$$v + u = \begin{bmatrix} v_1 + u_1 \\ v_2 + u_2 \\ \vdots \\ v_n + u_n \end{bmatrix}, \text{ and } v - u = \begin{bmatrix} v_1 - u_1 \\ v_2 - u_2 \\ \vdots \\ v_n - u_n \end{bmatrix}$$

Example:

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, u = \begin{bmatrix} -2 \\ 4 \\ 0 \\ 3 \end{bmatrix}, \text{ then } v + s = \begin{bmatrix} 1 + (-2) \\ 2 + 4 \\ 3 + 0 \\ 4 + 3 \end{bmatrix} = \begin{bmatrix} -1 \\ 6 \\ 3 \\ 7 \end{bmatrix}$$

Basic operations (vectors)

4. Norm

For $v = [v_1, v_2, \dots, v_n]$ a vector, its norm is defined as follows:

$$\|v\|_k = \sqrt[k]{\sum_{i=1}^n |v_i|^k}$$

Depending on the value of “k”, a norm can be:

- $k=1 \rightarrow$ Manhattan norm
- $k=2 \rightarrow$ Euclidian norm

All norms satisfy certain properties, such as non-negativity, the triangle inequality, and homogeneity.

Basic operations (vectors)

5. Distance

For $v = [v_1, v_2, \dots, v_n]$ a vector, and $u = [u_1, u_2, \dots, u_n]$ another vector, we define the distance between v and u as:

$$\text{dist}_k(v, u) = \|v - u\|_k$$

Depending on the value of “k”, a distance can be:

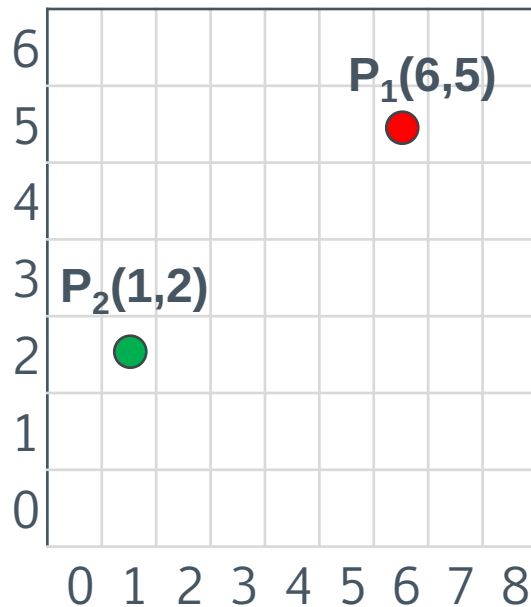
- $k=1 \rightarrow$ Manhattan norm
- $k=2 \rightarrow$ Euclidian norm

As a general consideration, dist_k is also called Minkovsky distance.

Basic operations (vectors)

5. Distance

Let's see an example (we will consider a 2-dimensional space).



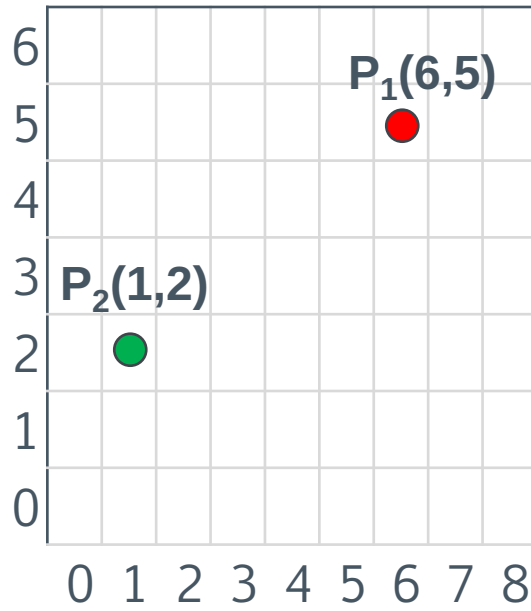
We can compute the distance based on a simple distance formula !

$$\begin{aligned} \text{dist}(P_1, P_2) &= \sqrt{(P_1.x - P_2.x)^2 + (P_1.y - P_2.y)^2} \\ &= \sqrt{(6 - 1)^2 + (5 - 2)^2} \\ &= \sqrt{25 + 9} = \sqrt{36} = 6 \end{aligned}$$

Basic operations (vectors)

5. Distance

Let's see an example (we will consider a 2-dimensional space).



Or we can consider P_1 and P_2 two bidimensional vectors and use the norm.

$$P_1 = \begin{bmatrix} 6 \\ 5 \end{bmatrix}, P_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \text{dist}(P_1, P_2) = \|P_1 - P_2\|_2$$

$$P_1 - P_2 = \begin{bmatrix} 6 - 1 \\ 5 - 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix},$$

$$\|P_1 - P_2\|_2 = \left\| \begin{bmatrix} 5 \\ 3 \end{bmatrix} \right\|_2 = \sqrt{5^2 + 3^2} = \sqrt{36} = 6$$

Basic operations (matrices)

1. Matrix transposition

Given a matrix “M” of shape n x m, the transpose of that matrix will be obtained by interchanging rows with columns:

$$M = \begin{bmatrix} m_{(1,1)} & \cdots & m_{(1,m)} \\ \vdots & \ddots & \vdots \\ m_{(n,1)} & \cdots & m_{(n,m)} \end{bmatrix}, \text{ then } M^T = \begin{bmatrix} m_{(1,1)} & \cdots & m_{(n,1)} \\ \vdots & \ddots & \vdots \\ m_{(1,m)} & \cdots & m_{(n,m)} \end{bmatrix}$$

Example:

$$M = \begin{bmatrix} 1 & 100 \\ 2 & 200 \\ 3 & 300 \end{bmatrix}, \text{ then } M^T = \begin{bmatrix} 1 & 2 & 3 \\ 100 & 200 & 300 \end{bmatrix}$$

Basic operations (matrices)

2. Matrix addition / subtraction

Given two matrixes “A” and “B” with the same size: n x m:

$$A = \begin{bmatrix} a_{(1,1)} & \cdots & a_{(1,m)} \\ \vdots & \ddots & \vdots \\ a_{(n,1)} & \cdots & a_{(n,m)} \end{bmatrix}, B = \begin{bmatrix} b_{(1,1)} & \cdots & b_{(1,m)} \\ \vdots & \ddots & \vdots \\ b_{(n,1)} & \cdots & b_{(n,m)} \end{bmatrix}, A + B = \begin{bmatrix} a_{(1,1)} + b_{(1,1)} & \cdots & a_{(n,1)} + b_{(1,m)} \\ \vdots & \ddots & \vdots \\ a_{(n,1)} + b_{(n,1)} & \cdots & a_{(n,m)} + b_{(n,m)} \end{bmatrix}$$

Example:

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, B = \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}, A + B = \begin{bmatrix} 1 + 7 & 4 + 10 \\ 2 + 8 & 5 + 11 \\ 3 + 9 & 6 + 12 \end{bmatrix} = \begin{bmatrix} 8 & 14 \\ 10 & 16 \\ 12 & 18 \end{bmatrix}$$

Basic operations (matrices)

3. Matrix multiplication

Given two matrixes “A” with size “ $n \times m$ ” and “B” with size “ $m \times k$ ” (meaning matrix “A” has the same number of columns as the number of rows from matrix “B”):

$$A = \begin{bmatrix} a_{(1,1)} & \cdots & a_{(1,m)} \\ \vdots & \ddots & \vdots \\ a_{(n,1)} & \cdots & a_{(n,m)} \end{bmatrix}, B = \begin{bmatrix} b_{(1,1)} & \cdots & b_{(1,p)} \\ \vdots & \ddots & \vdots \\ b_{(m,1)} & \cdots & b_{(m,p)} \end{bmatrix}, A \times B = C, \text{ with } C_{(i,j)} = \sum_{l=1}^m a_{(i,l)} \times b_{(l,j)}$$

Example (2 x 3 matrix with a 3 x 2 matrix):

$$A = \begin{bmatrix} a_{(1,1)} & a_{(1,2)} & a_{(1,3)} \\ a_{(2,1)} & a_{(2,2)} & a_{(2,3)} \end{bmatrix}, B = \begin{bmatrix} b_{(1,1)} & b_{(1,2)} \\ b_{(2,1)} & b_{(2,2)} \\ b_{(3,1)} & b_{(3,2)} \end{bmatrix},$$

$$A \times B = \begin{bmatrix} a_{(1,1)} \times b_{(1,1)} + a_{(1,2)} \times b_{(2,1)} + a_{(1,3)} \times b_{(3,1)} & a_{(1,1)} \times b_{(1,2)} + a_{(1,2)} \times b_{(2,2)} + a_{(1,3)} \times b_{(3,2)} \\ a_{(2,1)} \times b_{(1,1)} + a_{(2,2)} \times b_{(2,1)} + a_{(2,3)} \times b_{(3,1)} & a_{(2,1)} \times b_{(1,2)} + a_{(2,2)} \times b_{(2,2)} + a_{(2,3)} \times b_{(3,2)} \end{bmatrix}$$

Basic operations (matrices)

3. Matrix multiplication

Let's define two functions that can be applied over a matrix (**col** and **row** that create a vector out of a column or a row from a matrix.).

Let's consider the matrix A with size " $n \times m$ "

$$A = \begin{bmatrix} a_{(1,1)} & \cdots & a_{(1,m)} \\ \vdots & \ddots & \vdots \\ a_{(n,1)} & \cdots & a_{(n,m)} \end{bmatrix}, col_i(A) = \begin{bmatrix} a_{(1,i)} \\ a_{(2,i)} \\ \vdots \\ a_{(n,i)} \end{bmatrix}, row_i(A) = \begin{bmatrix} a_{(i,1)} \\ a_{(i,2)} \\ \vdots \\ a_{(i,m)} \end{bmatrix}$$

Basic operations (matrices)

3. Matrix multiplication

Then, we can write the matrix multiplication using the dot product of two vectors:

$$A = \begin{bmatrix} a_{(1,1)} & \cdots & a_{(1,m)} \\ \vdots & \ddots & \vdots \\ a_{(n,1)} & \cdots & a_{(n,m)} \end{bmatrix}, B = \begin{bmatrix} b_{(1,1)} & \cdots & b_{(1,p)} \\ \vdots & \ddots & \vdots \\ b_{(m,1)} & \cdots & b_{(m,p)} \end{bmatrix},$$

$$A \times B = C, \text{ with } C_{(i,j)} = \text{row}_i(A) \cdot \text{col}_j(B)$$

Example (2 x 3 matrix with a 3 x 2 matrix):

$$A = \begin{bmatrix} a_{(1,1)} & a_{(1,2)} & a_{(1,3)} \\ a_{(2,1)} & a_{(2,2)} & a_{(2,3)} \end{bmatrix}, B = \begin{bmatrix} b_{(1,1)} & b_{(1,2)} \\ b_{(2,1)} & b_{(2,2)} \\ b_{(3,1)} & b_{(3,2)} \end{bmatrix}, A \times B = \begin{bmatrix} \text{row}_1(A) \cdot \text{col}_1(B) & \text{row}_1(A) \cdot \text{col}_2(B) \\ \text{row}_2(A) \cdot \text{col}_1(B) & \text{row}_2(A) \cdot \text{col}_2(B) \end{bmatrix}$$

This concept will further be relevant to the study of **DENSE** neural networks.

History of neural networks

History

1943

Warren McCulloch and **Walter Pitts** introduced a mathematical model of a neural network, describing how neurons in the brain might work and proposing a simple model with thresholding units.

1957

Frank Rosenblatt introduced the *Perceptron*, one of the first algorithms to model neuron activation and learning

1960

Widrow and **Hoff** introduced *Adaline* (Adaptive Linear Neuron), which utilized the least mean squares algorithm in its learning phase

1960

Marvin Minsky and **Seymour Papert's** book "*Perceptrons*" highlighted the limitations of perceptrons, namely that they couldn't solve problems like the XOR problem.

History

1974

Paul Werbos is credited for being the first to introduce *backpropagation* for training neural networks. He described the method in his PhD. thesis, "Beyond Regression".

1975

Kunihiko Fukushima developed the *Cognitron* and later the *Neocognitron*. The Neocognitron, was an early type of convolutional neural network (CNN) and is seen as a predecessor to modern CNNs.

1982

John Hopfield introduced the *Hopfield Network*, a recurrent neural network (RNN) that could store patterns.

1986

David Rumelhart, **Geoffrey Hinton**, and **Ronald Williams** introduced the *backpropagation* algorithm for training multi-layer perceptrons (MLPs), paving the way for modern neural networks

History

1986

Parallel Distributed Processing (PDP) volumes were published by **David E. Rumelhart**, **James L. McClelland**, and **PDP Research Group**, discussing various foundational concepts and the value of distributed representations.

1989

LeNet, developed by **Yann LeCun** & co, is one of the earliest convolutional neural networks (CNNs) and played an important role in demonstrating the potential of such architectures for hand-written digit recognition.

1990

Jeff Elman introduced "Elman networks" - a type of recurrent neural network where context units are used to store the previous state of the hidden layer, allowing the network to capture temporal dependencies

1997

Sepp Hochreiter and **Jürgen Schmidhuber** introduce a new architectures: Long Short-Term Memory (LSTM) , a type of RNN designed to address the vanishing gradient problem.

History

2006

Geoffrey Hinton and his colleagues introduced Deep Belief Network. Their work showed that deep architectures could be pre-trained layer-by-layer using Restricted Boltzmann Machines and then fine-tuned using backpropagation.

2009

A landmark paper titled "Large-scale Deep Unsupervised Learning using Graphics Processors" was presented by **Rajat Raina**, **Anand Madhavan**, and **Andrew Ng** explaining the value of GPUs in training neural networks.

2009

The first large dataset was made available for research: *ImageNet*.

2010

One of the most significant milestones in the field of neural networks and deep learning was the introduction of the *Rectified Linear Unit (ReLU)* activation function for deep neural networks

History

2012

Deep Convolutional Networks was introduced by **Alex Krizhevsky**, **Ilya Sutskever**, and **G. Hinton** at the ImageNet Large Scale Visual Recognition Challenge, setting off a deep learning revolution in computer vision.

2014

Ian Goodfellow and his colleagues introduced GANs (Generative Adversarial Networks), providing a novel way to generate synthetic data by pitting two neural networks against each other.

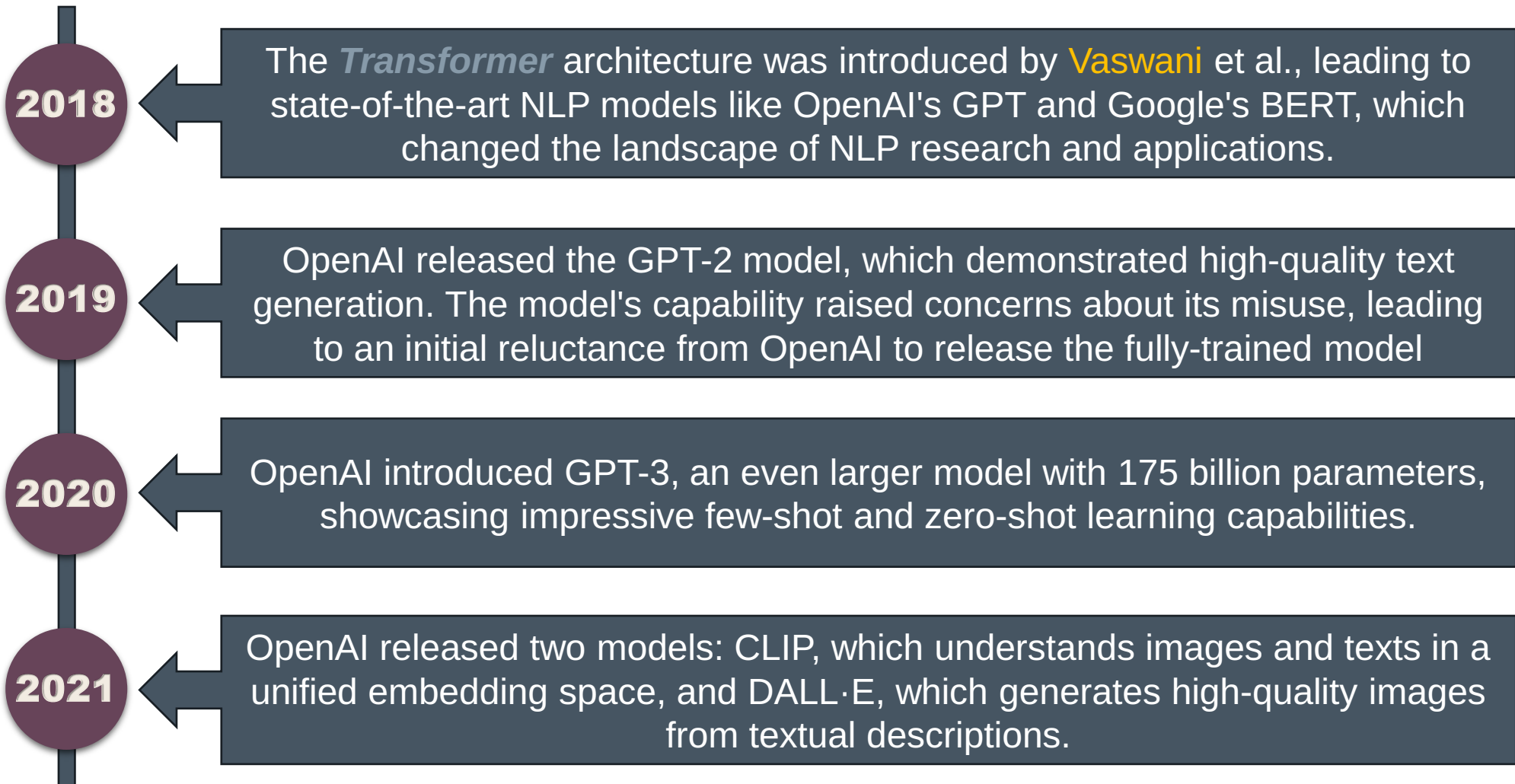
2014

Ilya Sutskever, **Oriol Vinyals**, and **Quoc V. Le** introduced the seq2seq framework (*Sequence-to-Sequence Learning*), revolutionizing tasks in natural language processing, especially machine translation.

2016

DeepMind's AlphaGo, which combined deep learning and Monte Carlo Tree Search, defeated the world champion Go player Lee Sedol, showcasing AI's capabilities in complex games

History



History

2021

DeepMind published its work on MuZero in Nature in January 2021. MuZero can master games like Chess, Shogi, Go, and Atari without knowing their rules in advance.

2022

OpenAI released Codex, which powered GitHub's Copilot, providing coding assistance by auto-completing lines or even entire functions.

π

Q & A

