# Introduction to

# Artificial Neural Networks

# and Deep Learning

*Nicolas Gambardella*

*nicolas.gambardella@univ-lille.fr*


Google


HD-03

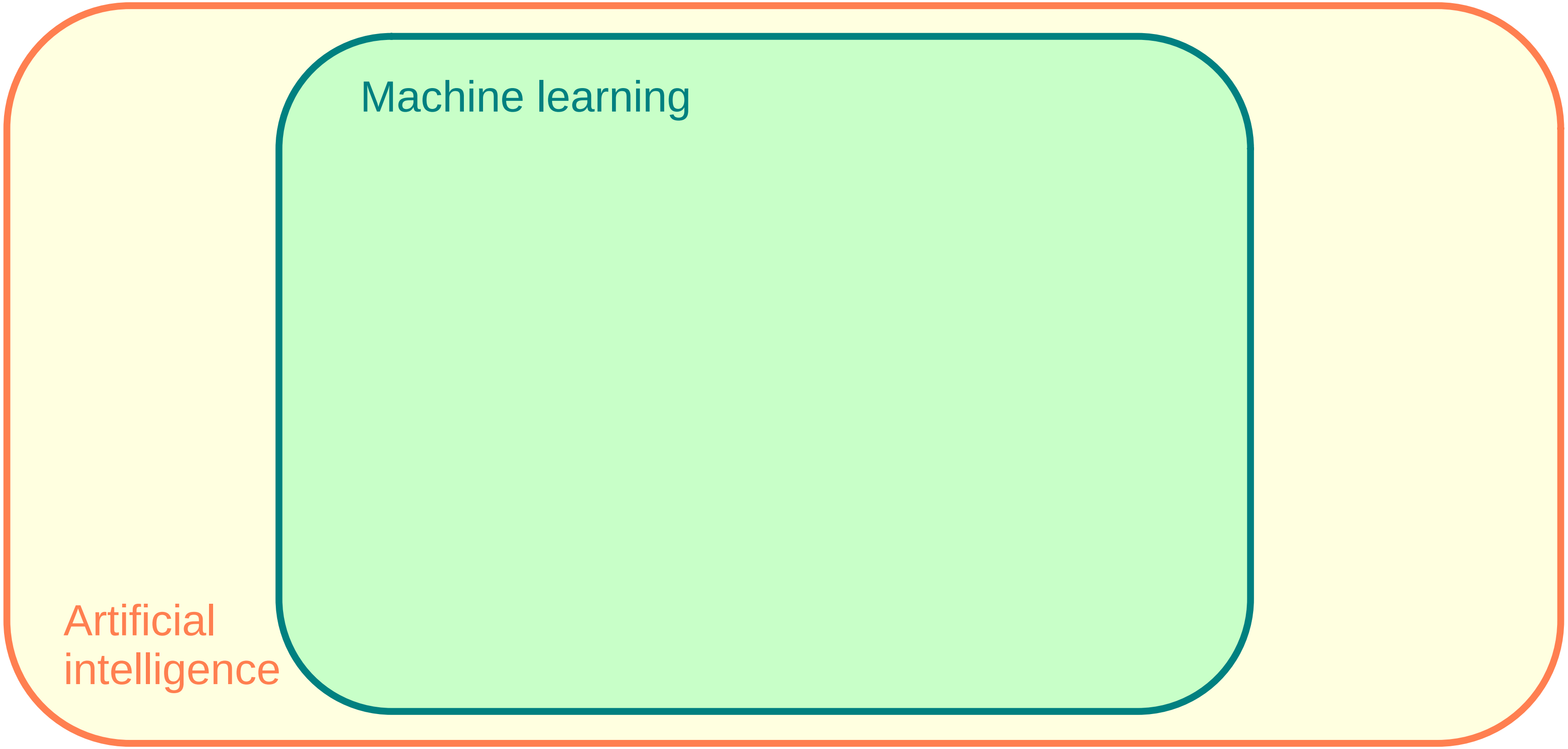

ChatGPT
Create an image for my presentation
Suggest a recipe based on a photo of my fridge
Create a workout plan
Write a report based on my data
Message ChatGPT
ChatGPT can make mistakes. Check important info.


AlphaGo
Lee Sedol

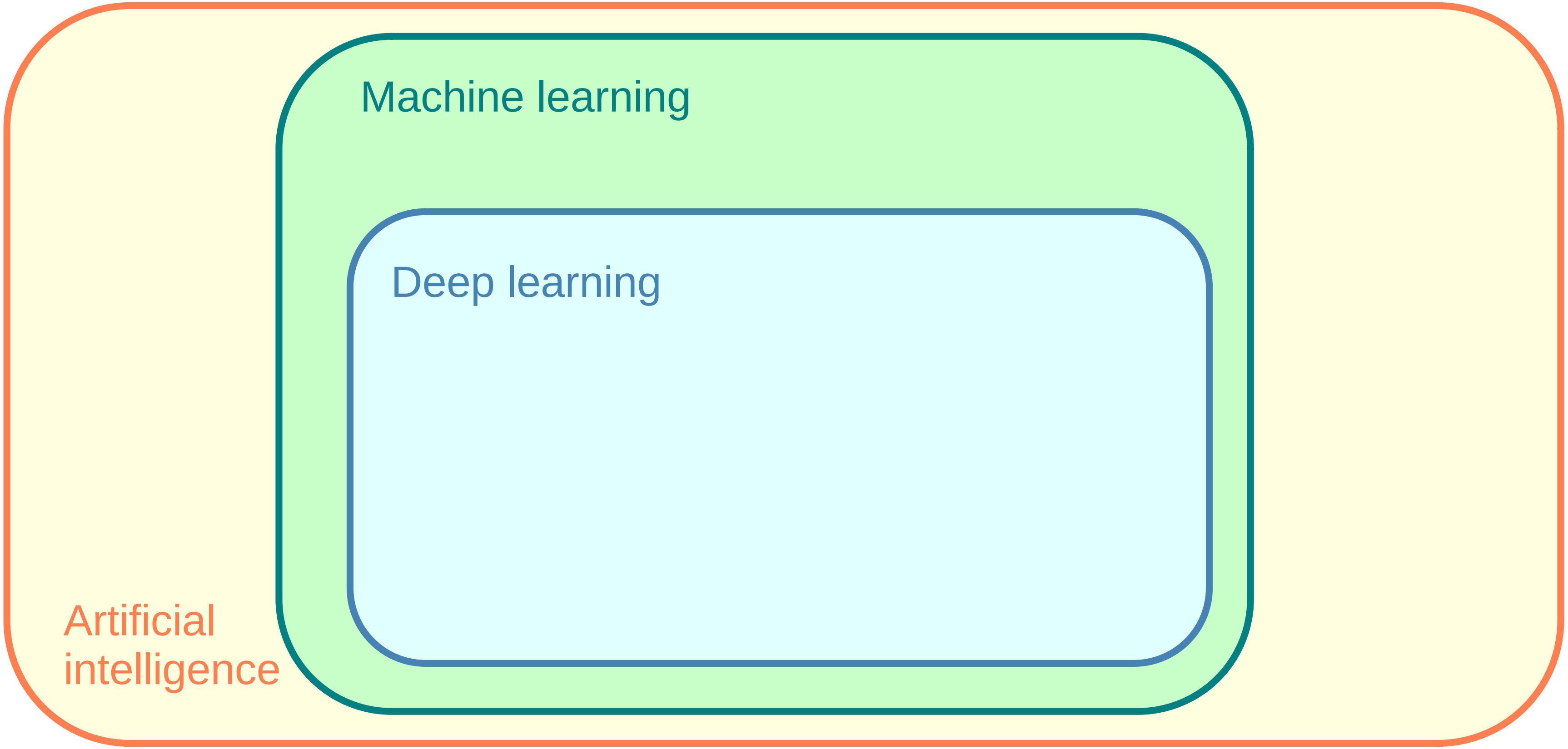

AlphaFold

# Some terminology

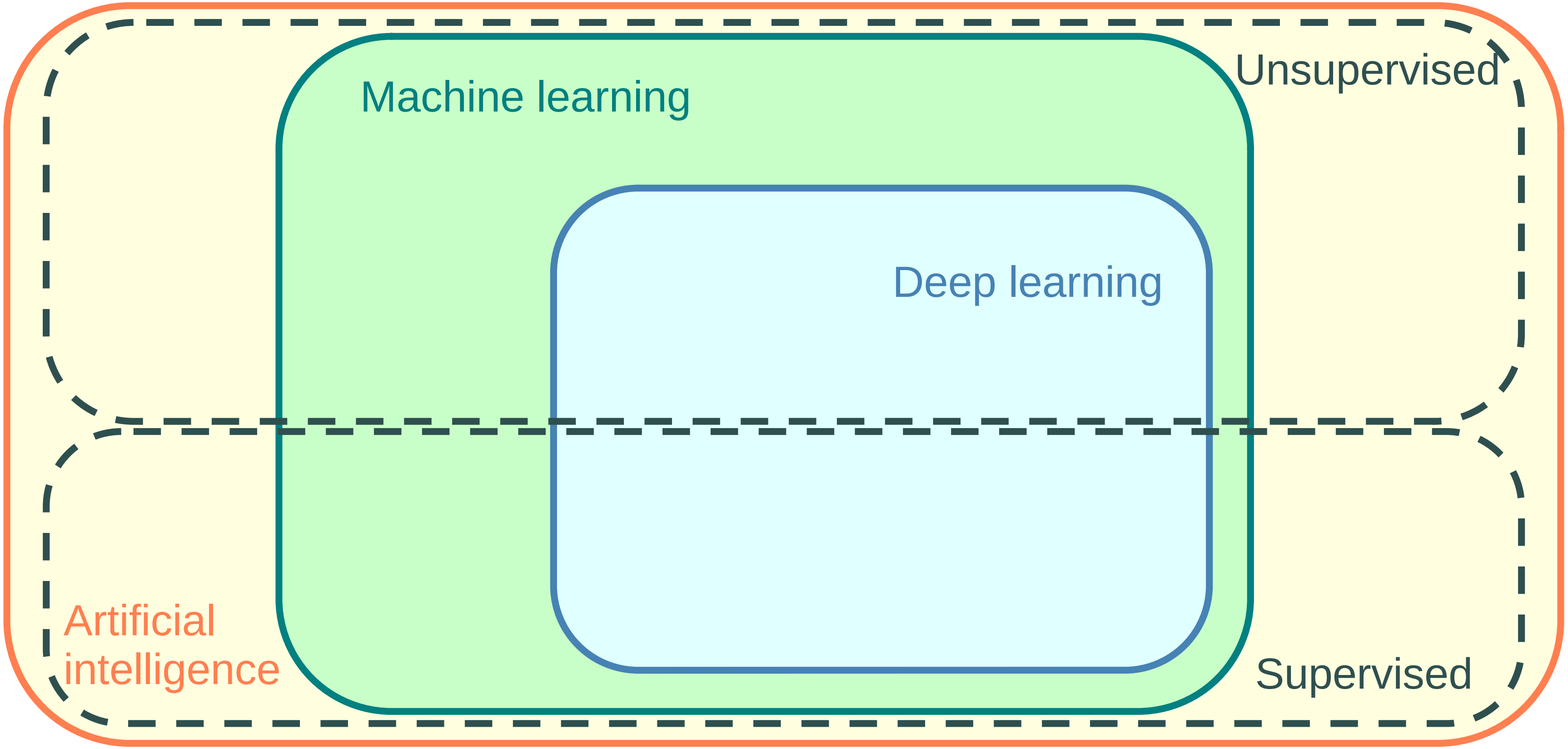
# Some terminology



# Some terminology

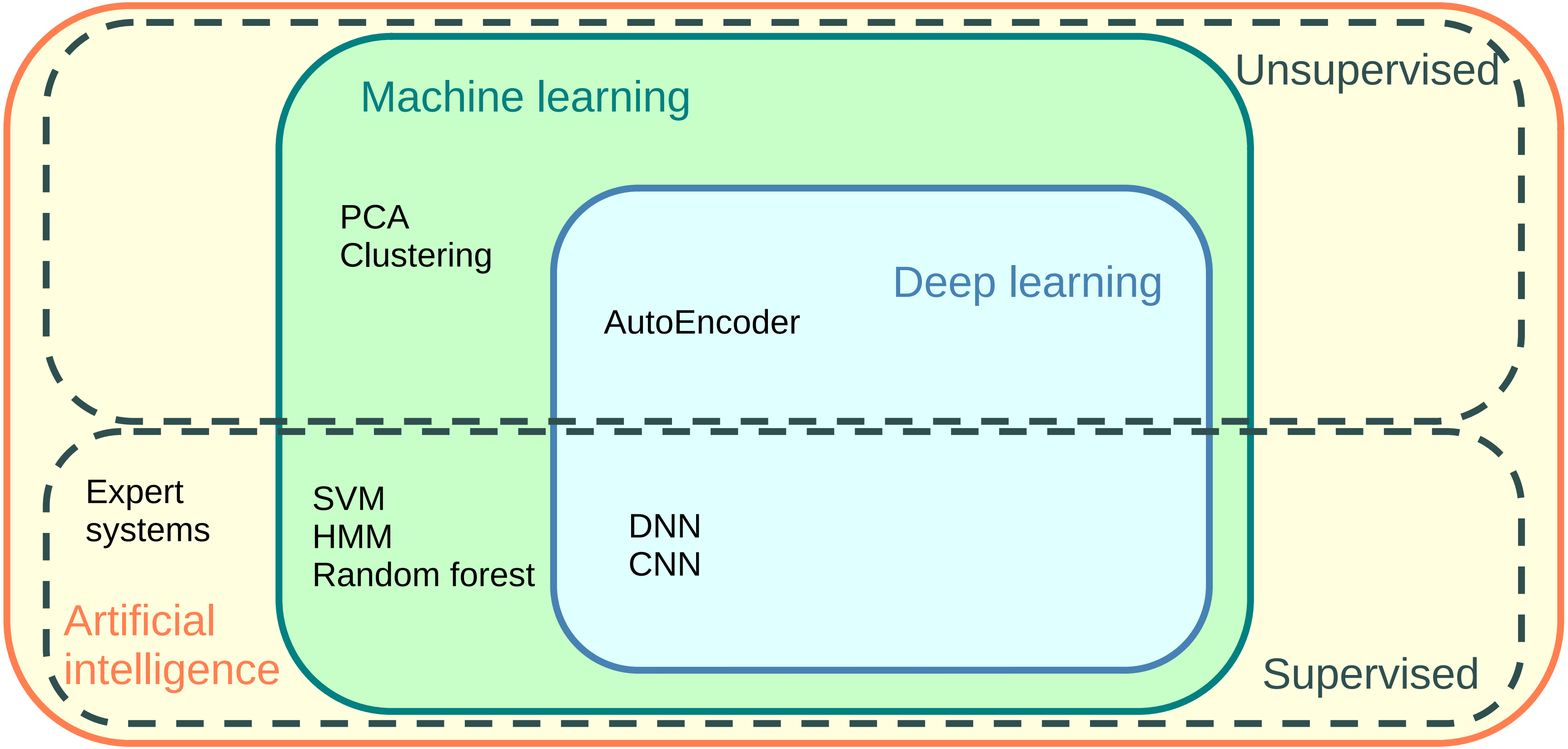


# Some terminology

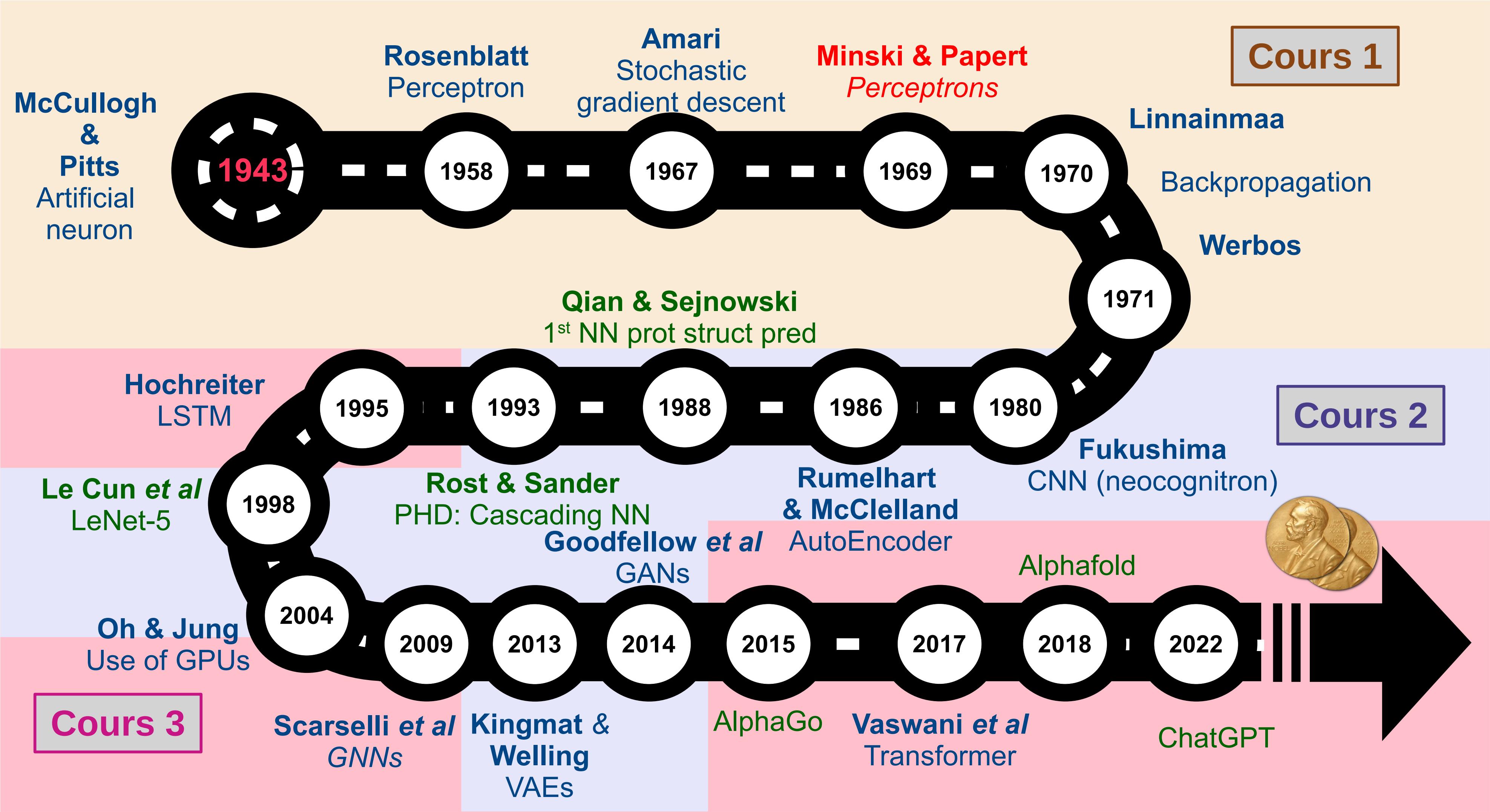



(NB: I consider reinforcement learning as part of supervised, but this is controversial)

# Some terminology




(NB: I consider reinforcement learning as part of supervised, but this is controversial)

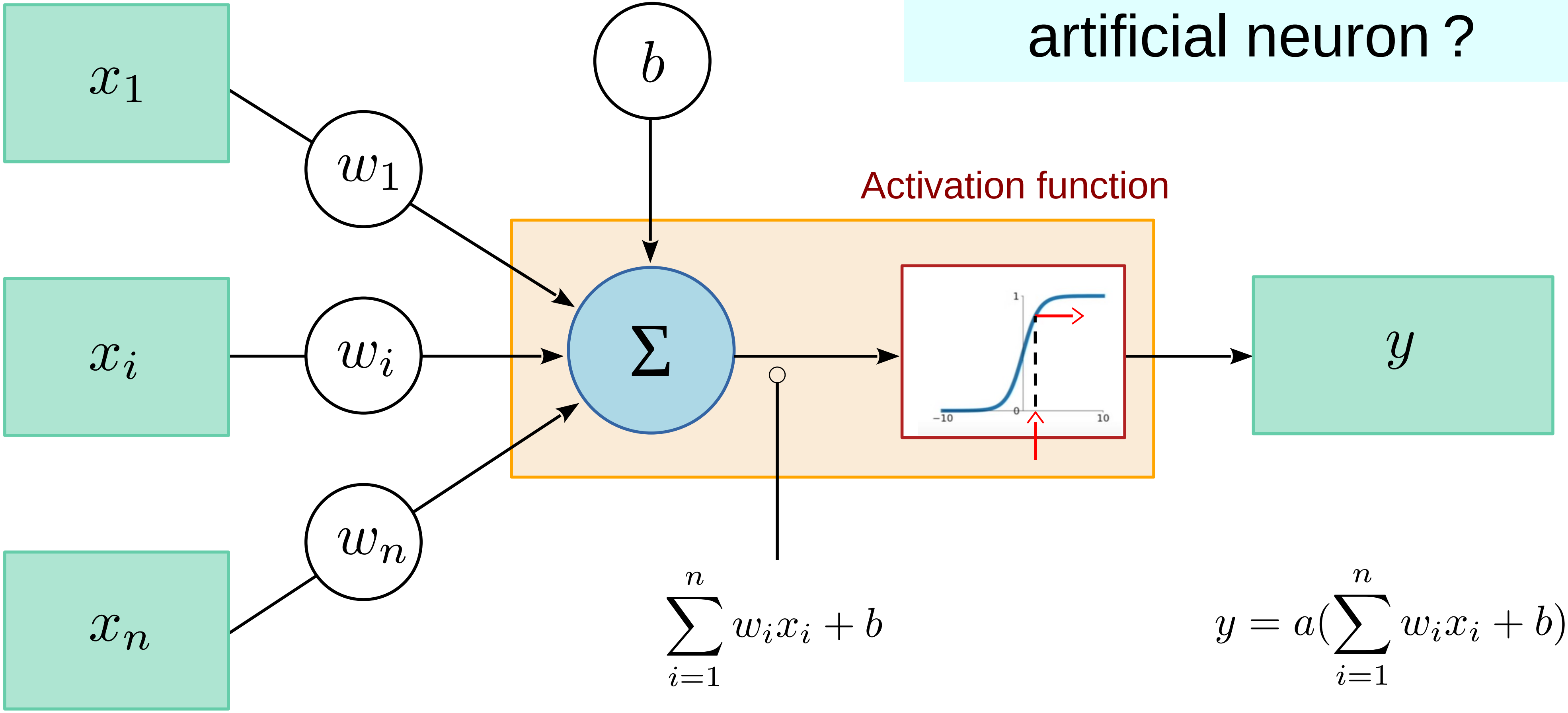


Cours 1
Cours 2
Cours 3
1943
McCullogh & Pitts
Artificial neuron
1958
Rosenblatt
Perceptron
1967
Amari
Stochastic gradient descent
1969
Minski & Papert
Perceptrons
1970
Linnainmaa
Backpropagation
1971
Werbos
1980
Fukushima
CNN (neocognitron)
1986
Rumelhart & McClelland
AutoEncoder
1988
Qian & Sejnowski
1st NN prot struct pred
1993
Rost & Sander
PHD: Cascading NN
1995
Hochreiter
LSTM
1998
Le Cun et al
LeNet-5
2004
Oh & Jung
Use of GPUs
2009
Scarselli et al
GNNs
2013
Kingmat & Welling
VAEs
2014
Goodfellow et al
GANs
2015
AlphaGo
2017
Vaswani et al
Transformer
2018
Alphafold
2022
ChatGPT

# What is an ANN ?

- A method to represent any possible equation
- A method to produce any possible curve in any number of dimensions
- A tool that can learn to recognize spatio-temporal patterns

# What is an artificial neuron ?

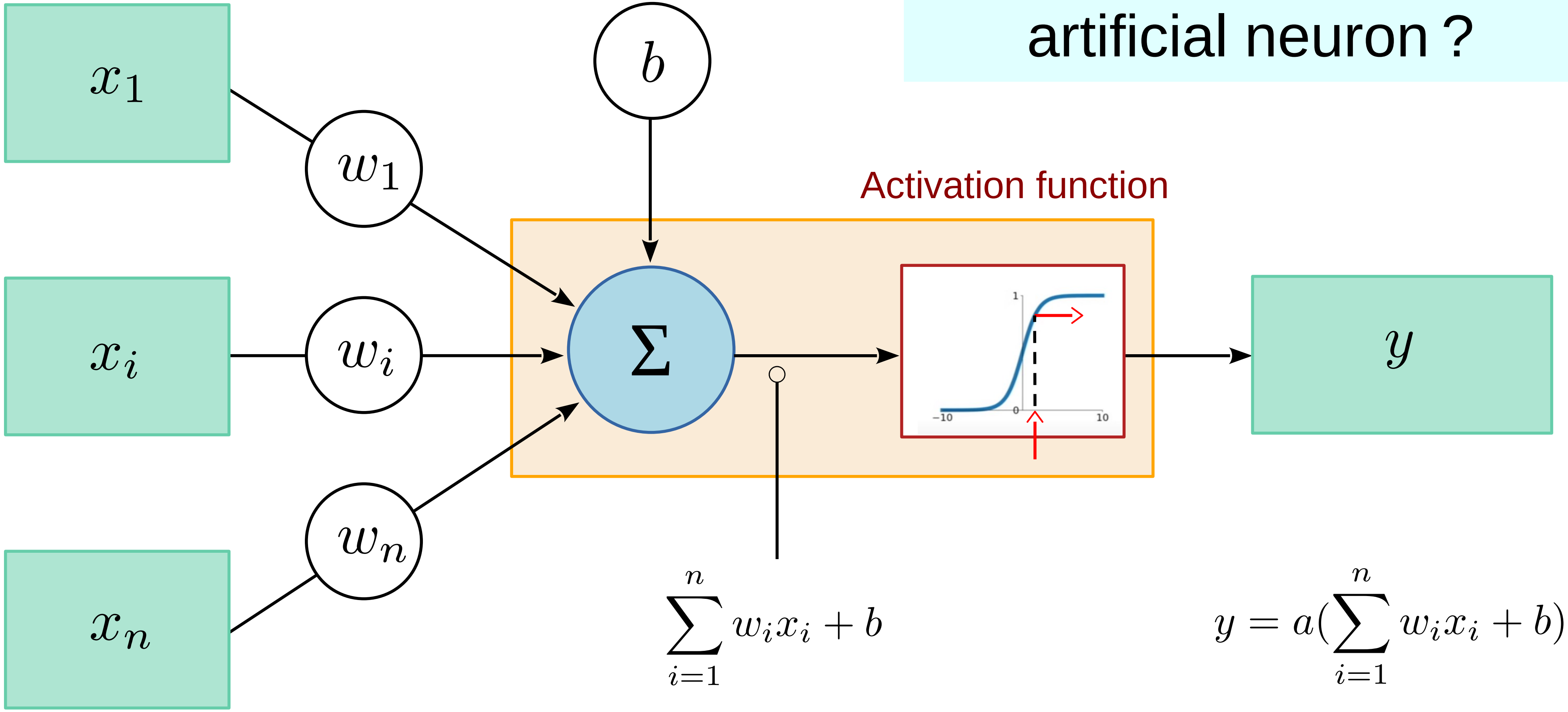



$$\sum_{i=1}^{n} w_i x_i + b$$

$$y = a(\sum_{i=1}^{n} w_i x_i + b)$$

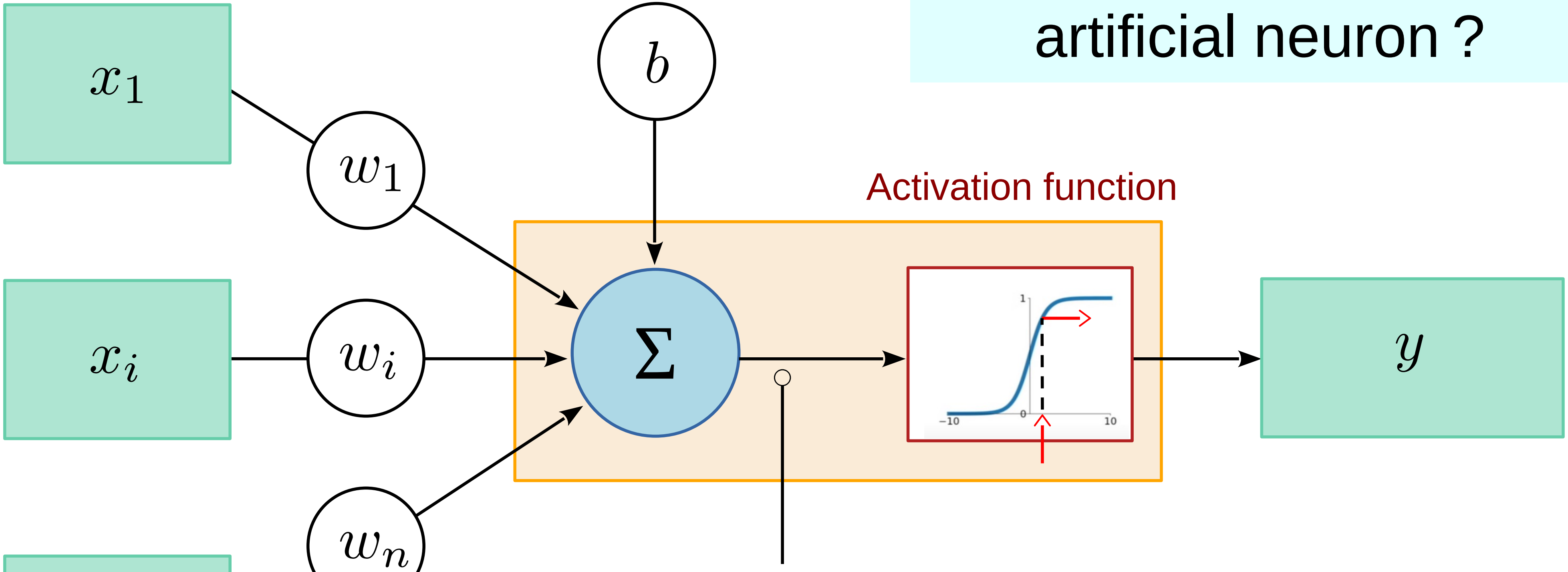
McCulloch and Pitts (1943) A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 5:115-133
Rosenblatt (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65(6):386-408
Widrow and Hoff (1960) Adaptive Switching circuits. *WESCON Convention record* part IV: 96-104

# What is an artificial neuron ?




NB: when the activation function is logistic (sigmoid), this is actually a logistic regression...

# What is an artificial neuron ?



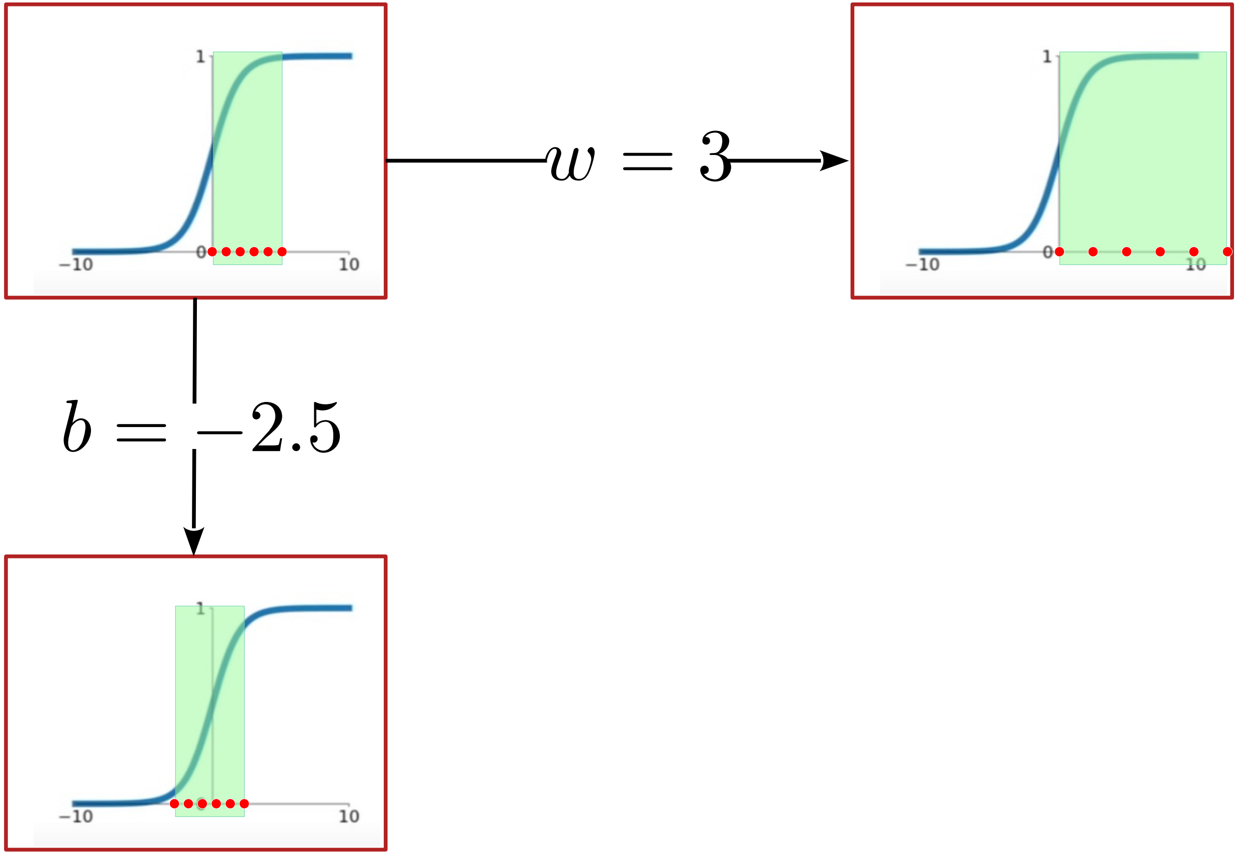

$$\sum_{i=1}^{n} w_i x_i + b = z$$

$$y = a(z)$$

NB: when the activation function is logistic (sigmoid), this is actually a logistic regression...

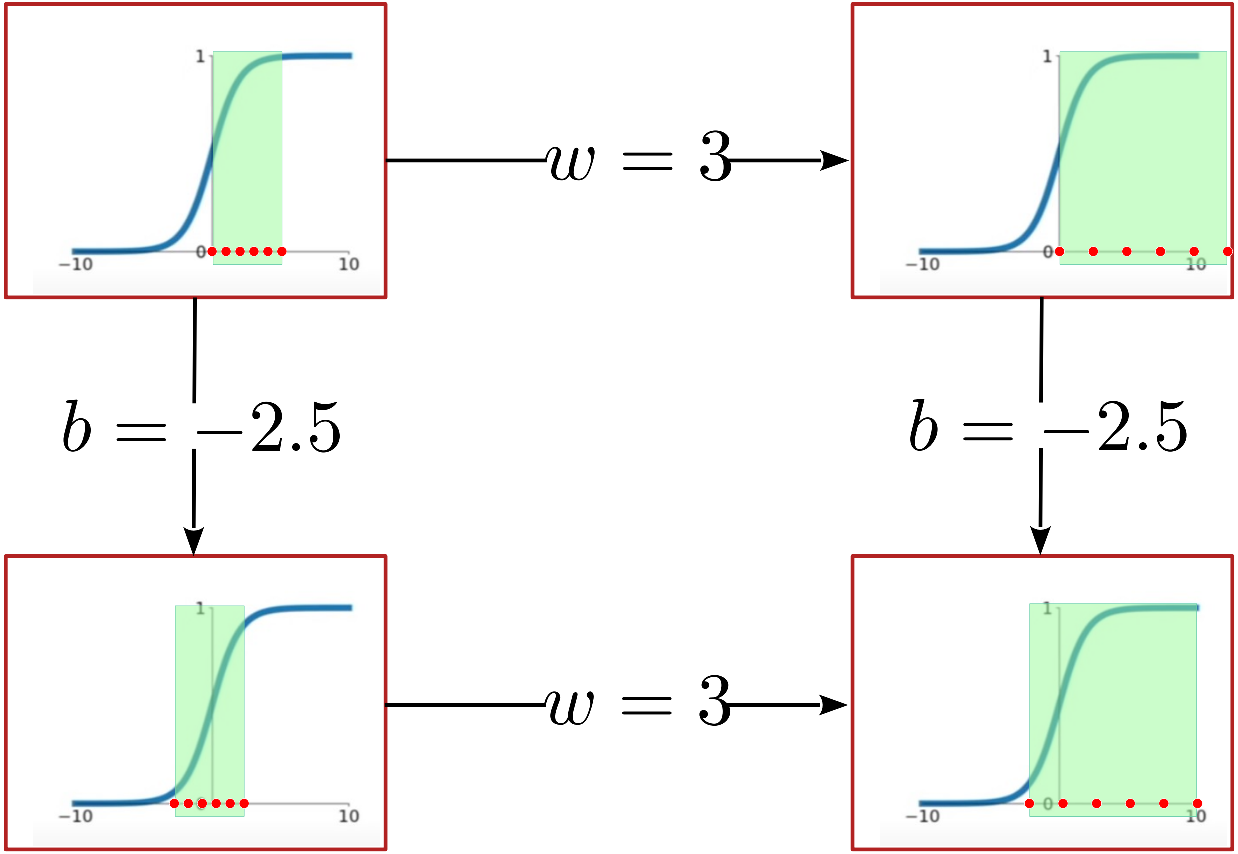
# Impact of the weights and the bias

inputs = [0,1,2,3,4,5]

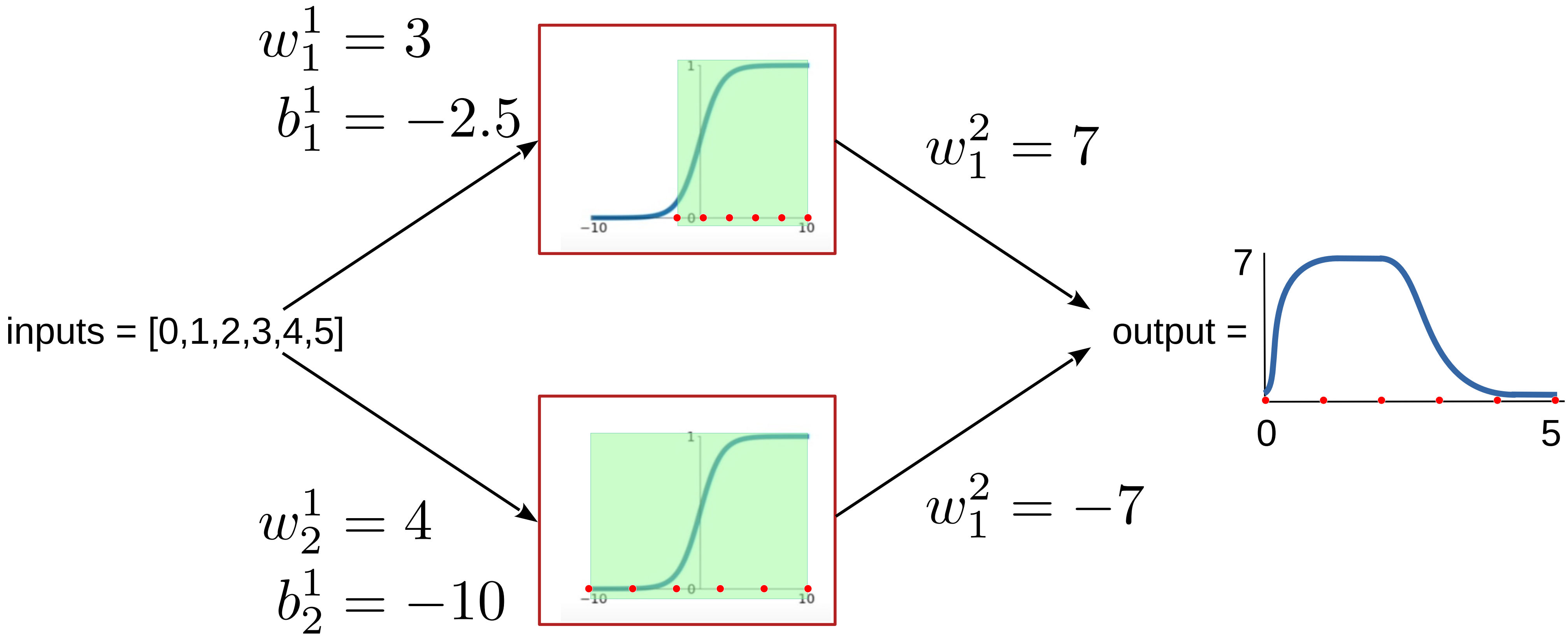


# Impact of the weights and the bias

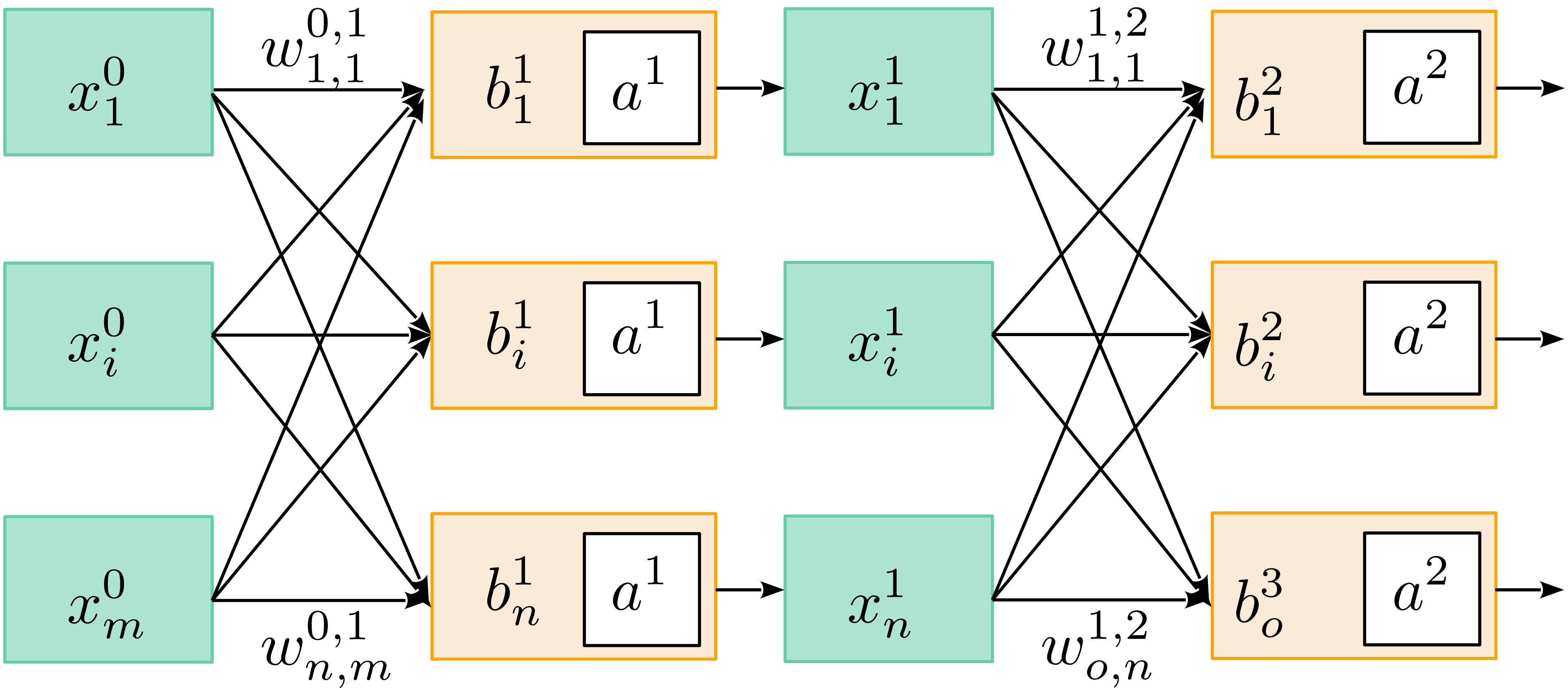
inputs = [0,1,2,3,4,5]



# The magic happens with several neurons

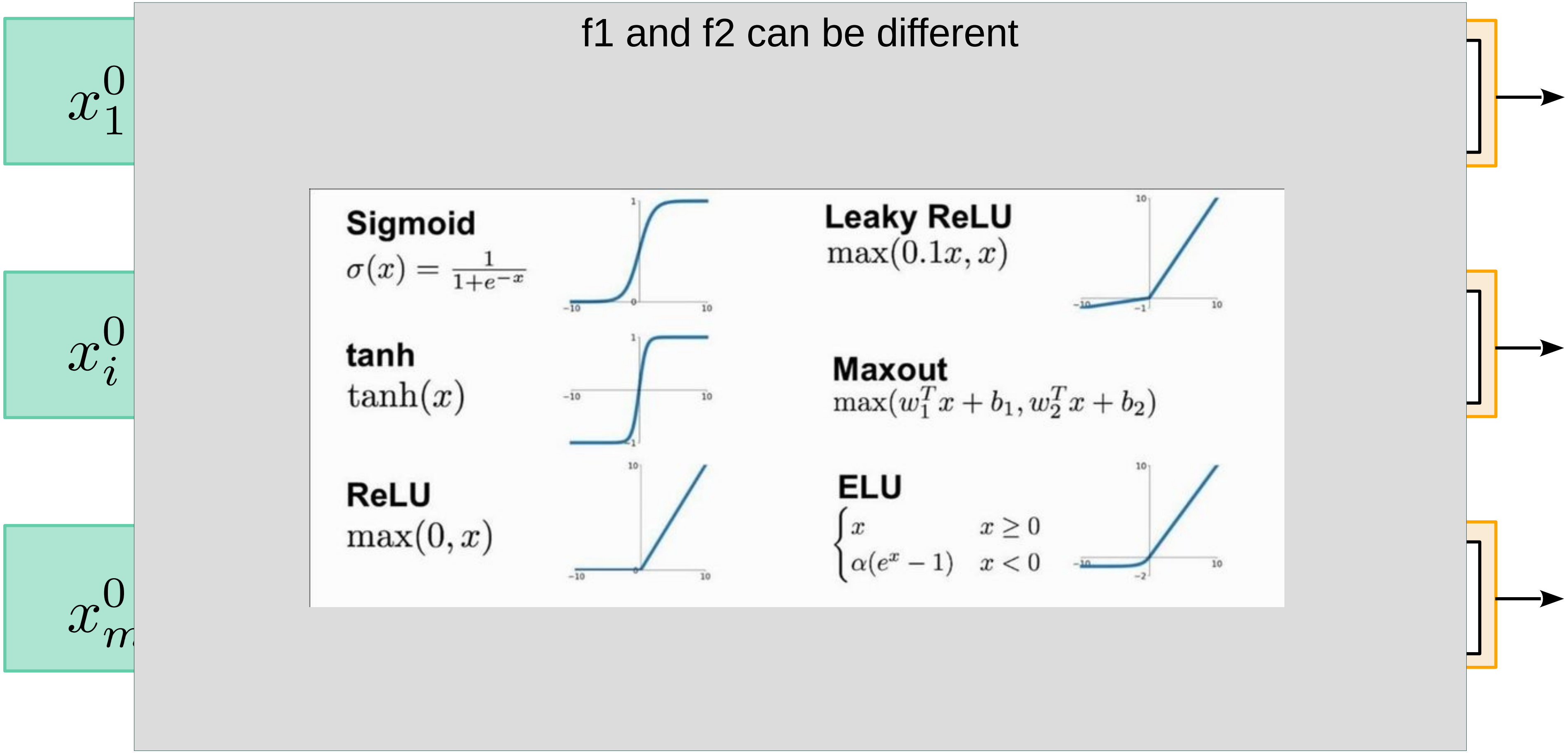


# And then we add layers (the "Deep")




Rosenblatt (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65(6):386-408

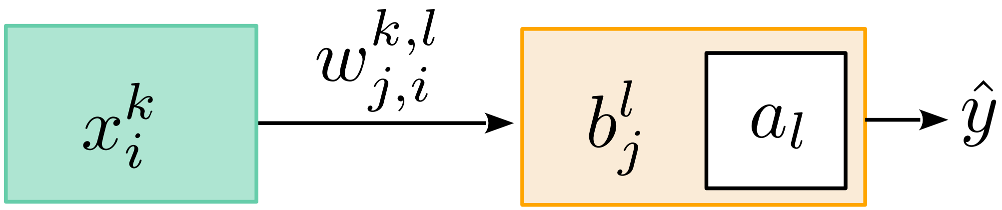
# And then we add layers (the “Deep”)



# A word on standard notation

$k$ = source layer

$l$ = target layer



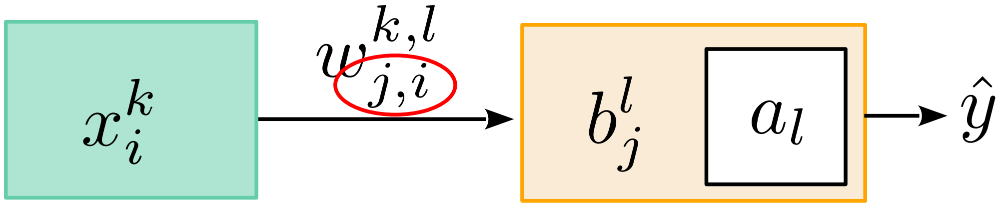

$i$ = source neuron

$j$ = target neuron

# A word on standard notation

$k$ = source layer

$l$ = target layer



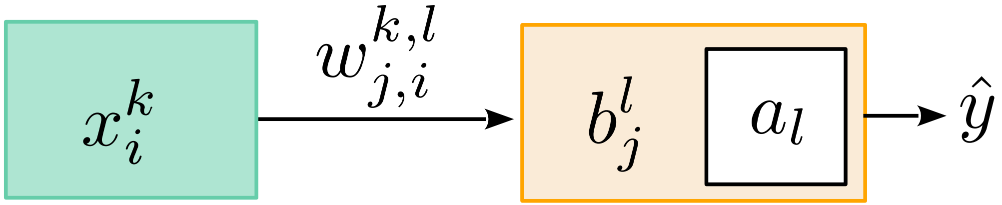

why???

$i$ = source neuron

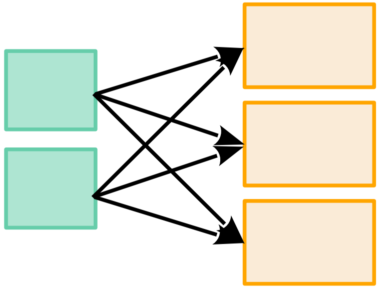
$j$ = target neuron

# A word on standard notation

$k =$ source layer $\qquad l =$ target layer



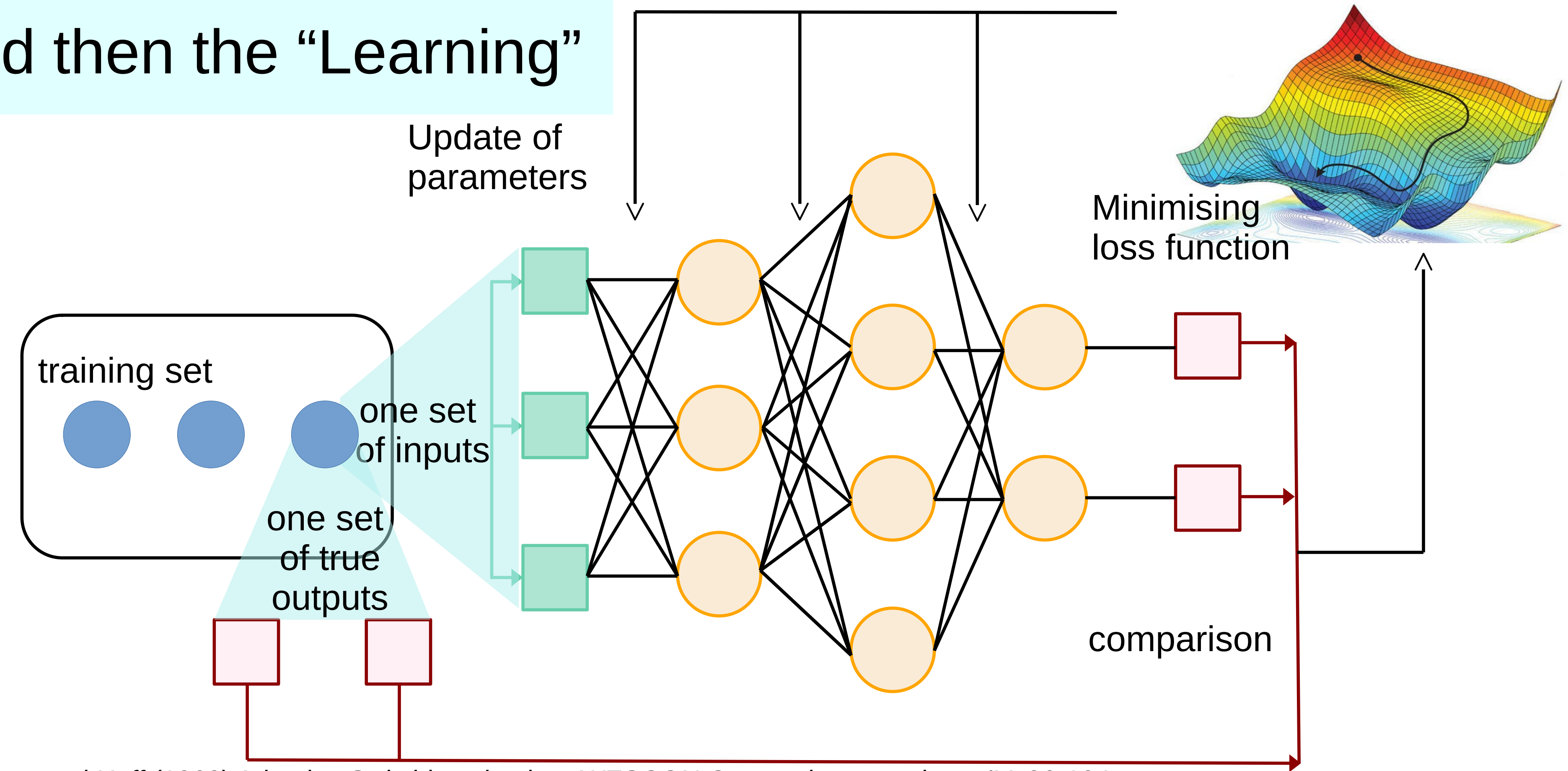

$i =$ source neuron $\qquad j =$ target neuron



$$\begin{pmatrix} \mathrm{in}_1 \\ \mathrm{in}_2 \\ \mathrm{in}_3 \end{pmatrix} = \begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 w_{1,1} + x_2 w_{1,2} \\ x_1 w_{2,1} + x_2 w_{2,2} \\ x_1 w_{3,1} + x_2 w_{4,2} \end{pmatrix}$$

Each portion of an ANN is stored as a multidimensional array: a tensor
There are tensors of numbers and also tensors of functions connecting other tensors

# And then the “Learning”




Widrow and Hoff (1960) Adaptive Switching circuits. *WESCON Convention record* part IV: 96-104
S Amari (1967). A theory of adaptive pattern classifier. *IEEE Transactions*. EC (16): 279–307
S Linnainmaa (1970-1976). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (Masters). University of Helsinki. p. 6–7.
P Werbos (1971-1982) Applications of advances in non-linear sensitivity analysis. *LNCIS* 38: 762-770
LeCun Y (1985) Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proc Cognitiva* 85, 599-604.
Rumelhart, Hinton, and Williams (1986) Learning representations by back-propagating errors." *Nature* 323(6088): 533-536.

# Parameters vs hyperparameters in DL

*Parameters* are learned: weights and biases
Models can have a dozen to several hundreds billions parameters

*Hyperparameters* are set

Architecture: # layers, # neurons, type of activation functions and regularisations

Evaluation: Loss function, optimizer, learning rate, decay

Training duration: Epochs, batch size, early stop

# Optimization, e.g., gradient descent

## Objective: to minimize a *loss function*
(*cost function* in optimization)

### Regression:

Number of samples

$$\mathrm{MSE} = \frac{1}{S} \sum_{i=1}^{\boxed{S}} (\boxed{y_{(i)}} - \boxed{\hat{y_{(i)}}})^2$$

True value

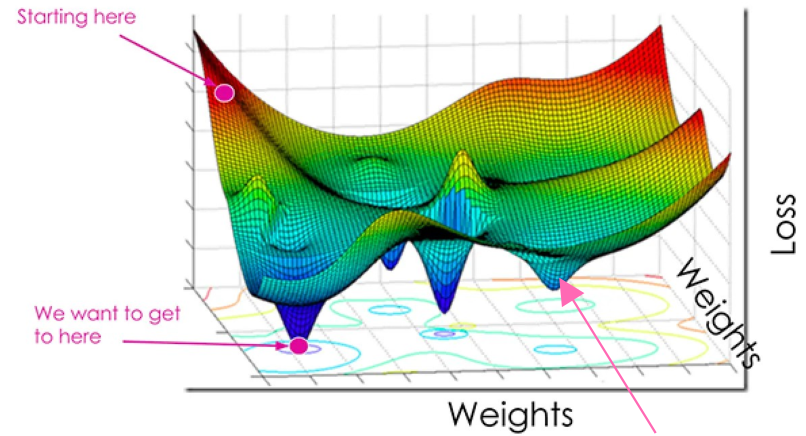
Predicted value

$$\mathrm{MAE} = \frac{1}{S} \sum_{i=1}^{S} \left| y_{(i)} - \hat{y_{(i)}} \right|$$

### Classification:

$$\mathrm{BCE} = -\frac{1}{S} \sum_{i=1}^{S} y_{(i)} \log(\hat{y_{(i)}}) + (1 - y_{(i)}) \log(1 - \hat{y_{(i)}})$$

$$\mathrm{CCE} = -\frac{1}{S} \sum_{i=1}^{S} \sum_{j=1}^{C} y_{(i)j} \log(\hat{y_{(i)}}_j)$$

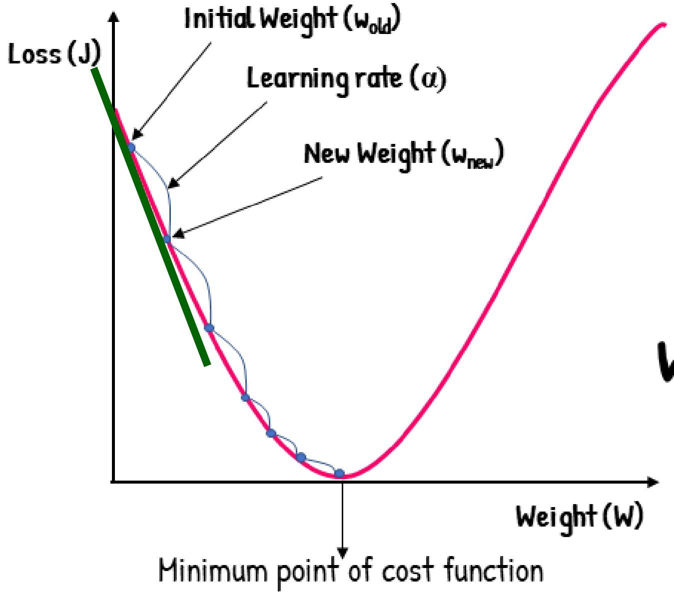



But not here!

MSE: mean squared error
MAE: mean averaged error
BCE: binary cross-entropy
CCE: categorical cross-entropy

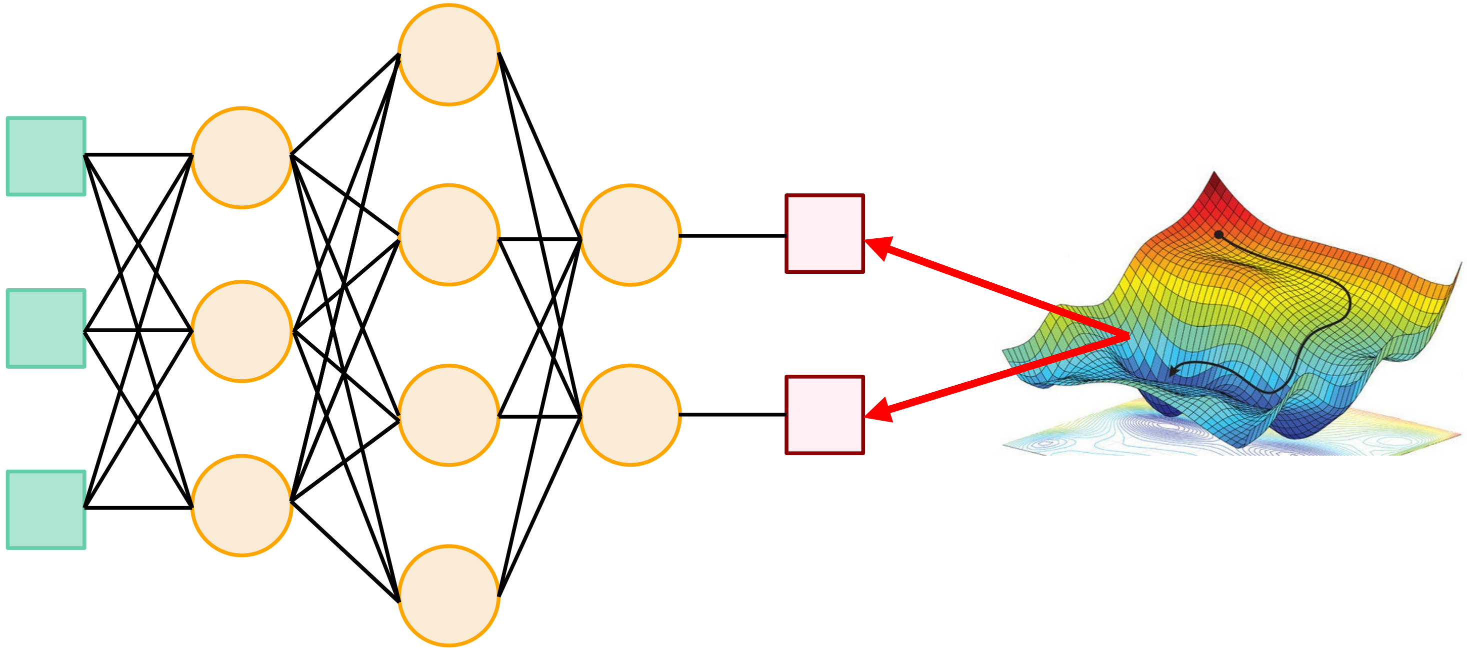
# Optimization, e.g., gradient descent

To minimise the loss function (called L, C, or most often J), we will calculate the “gradient” – the derivative – of the loss function with a set of parameters and calculate a new set using this gradient and the *learning rate*.

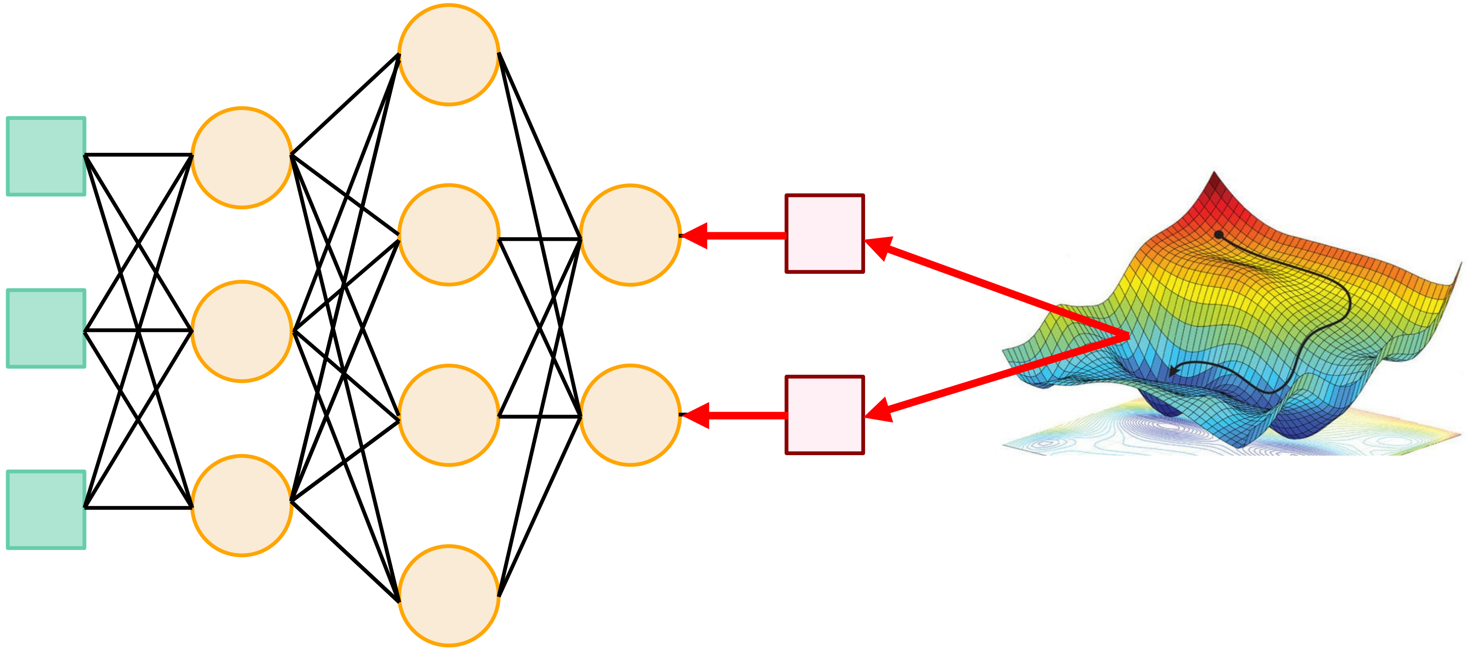



$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

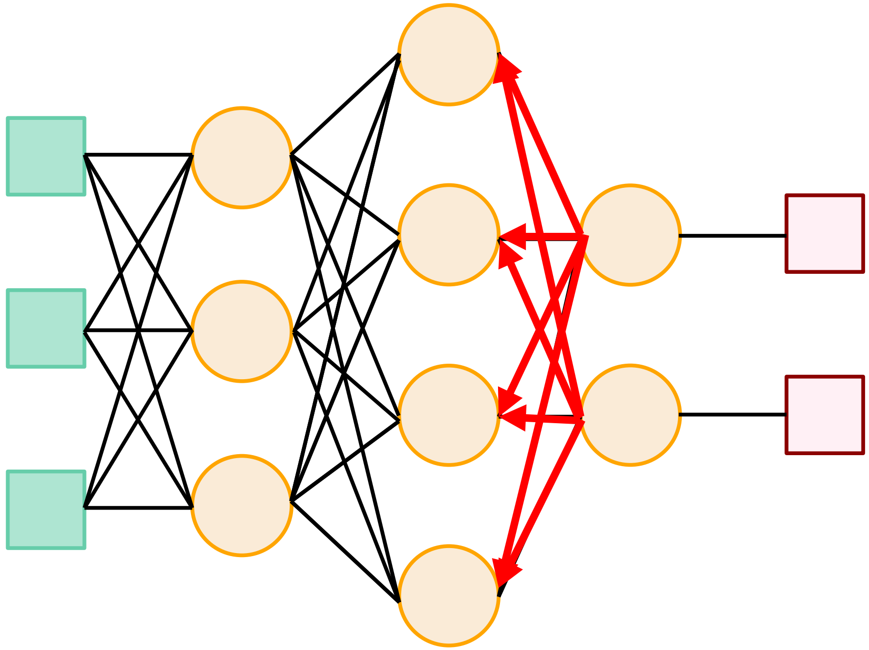
# Error backpropagation one layer at a time

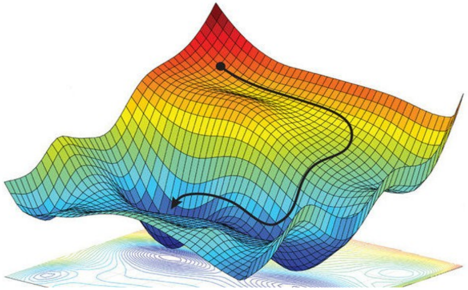


# Error backpropagation one layer at a time

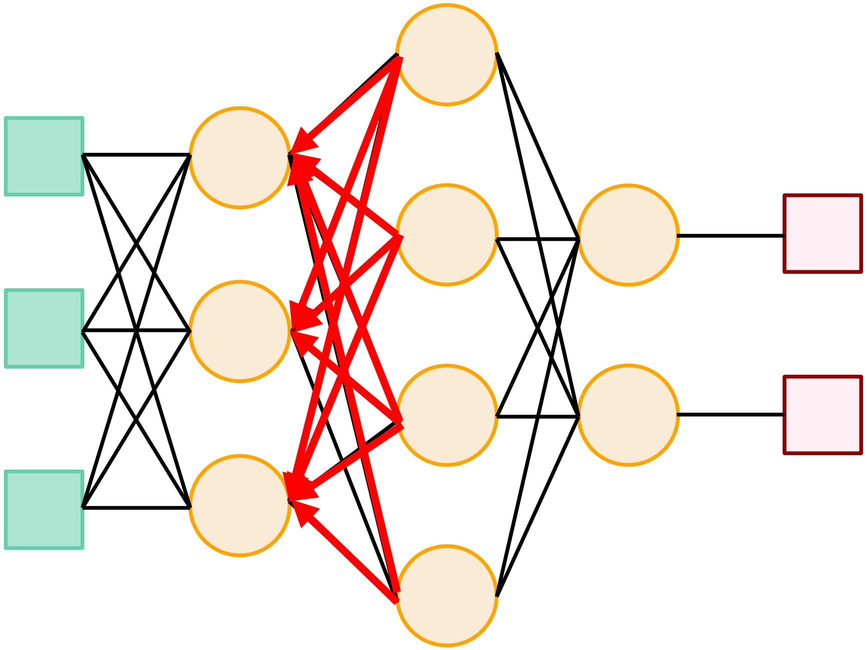


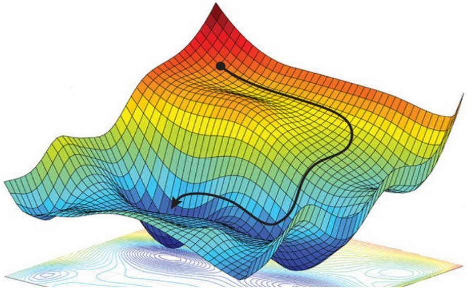
# Error backpropagation one layer at a time



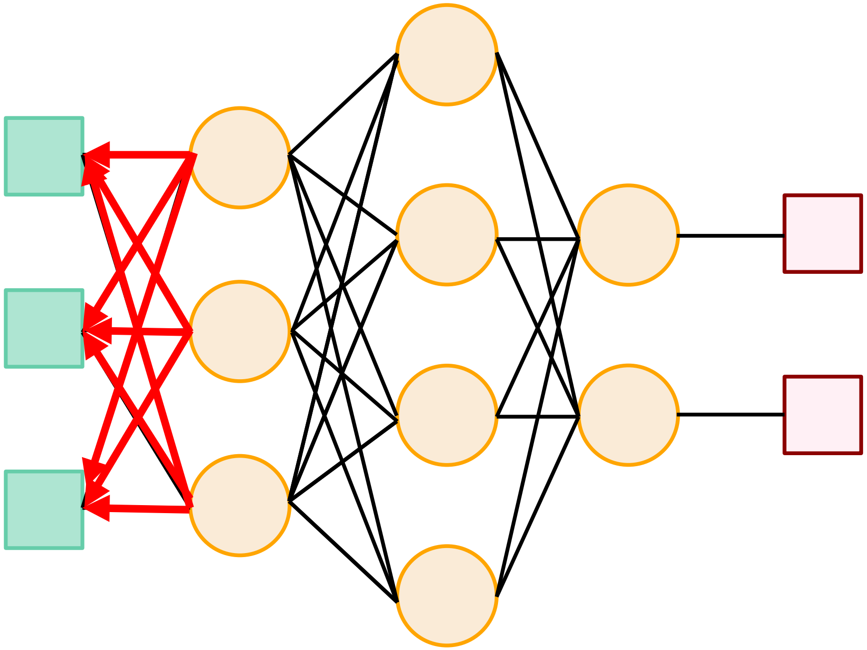


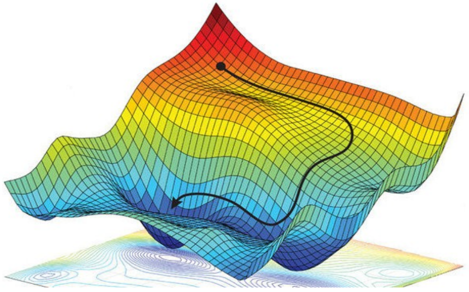
# Error backpropagation one layer at a time



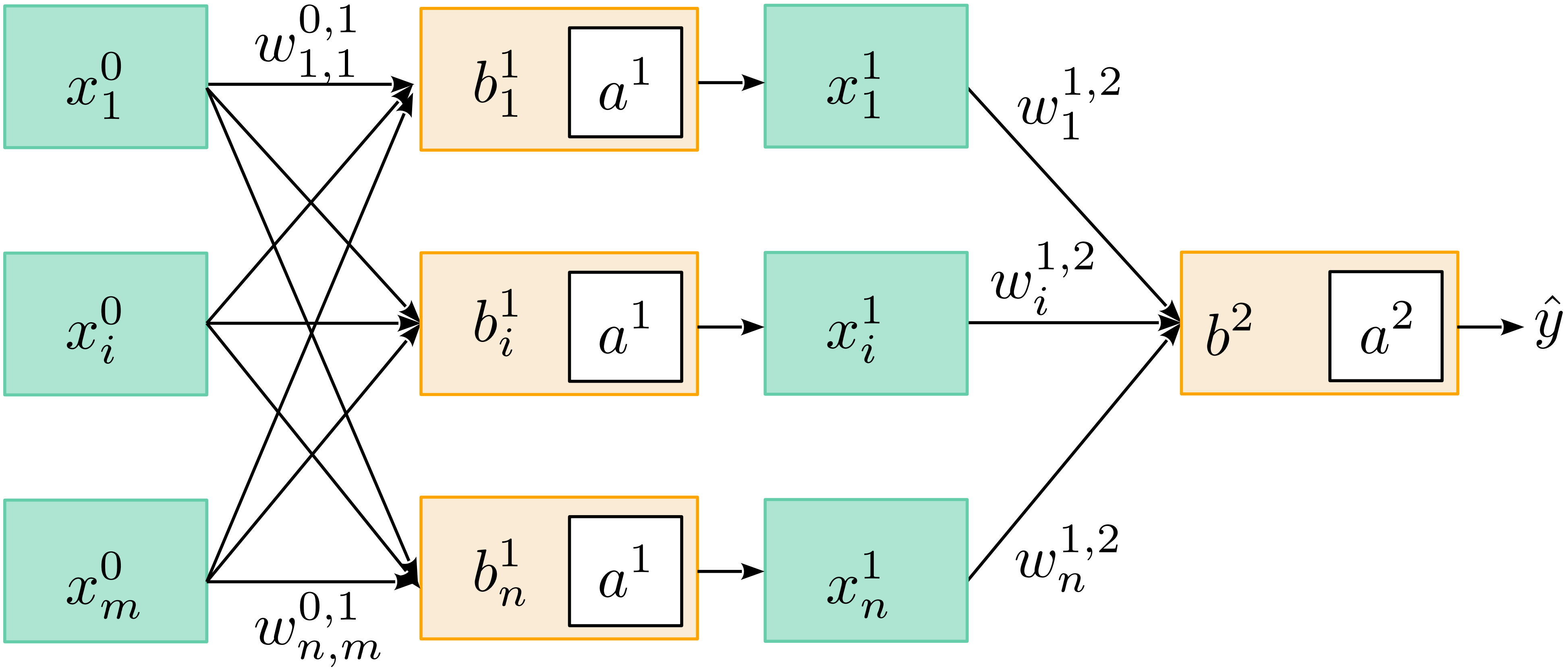


# Error backpropagation one layer at a time





# Learning: Backpropagation



# Backpropagation

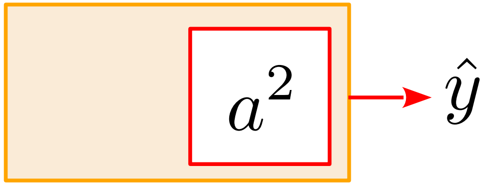
$$\mathrm{MSE} = \frac{1}{S}\sum_{s=1}^{S}(y(s) - y\hat{(s)})^2$$

$$\longrightarrow \hat{y}$$

# Backpropagation

$$\text{MSE} = \frac{1}{S}\sum_{s=1}^{S}(y(s) - y(\hat{s}))^2$$

$$\hat{y} = a^2(z^2) \qquad \text{e.g.} \qquad \frac{1}{1+e^{-z^2}}$$

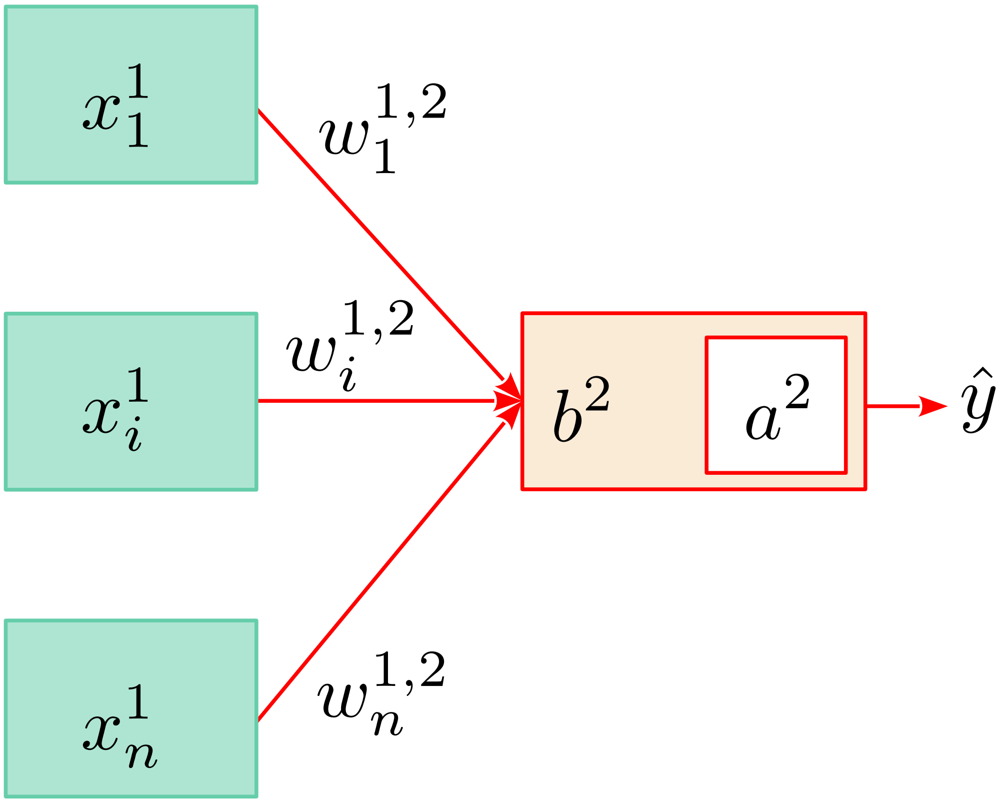


# Backpropagation

$$\text{MSE} = \frac{1}{S}\sum_{s=1}^{S}(y(s) - \widehat{y(s)})^2$$

$$\hat{y} = a^2(z^2)$$

$$z^2 = \sum_{i}^{n} w_i^{1,2} x_i^1 + b^2$$



# Backpropagation

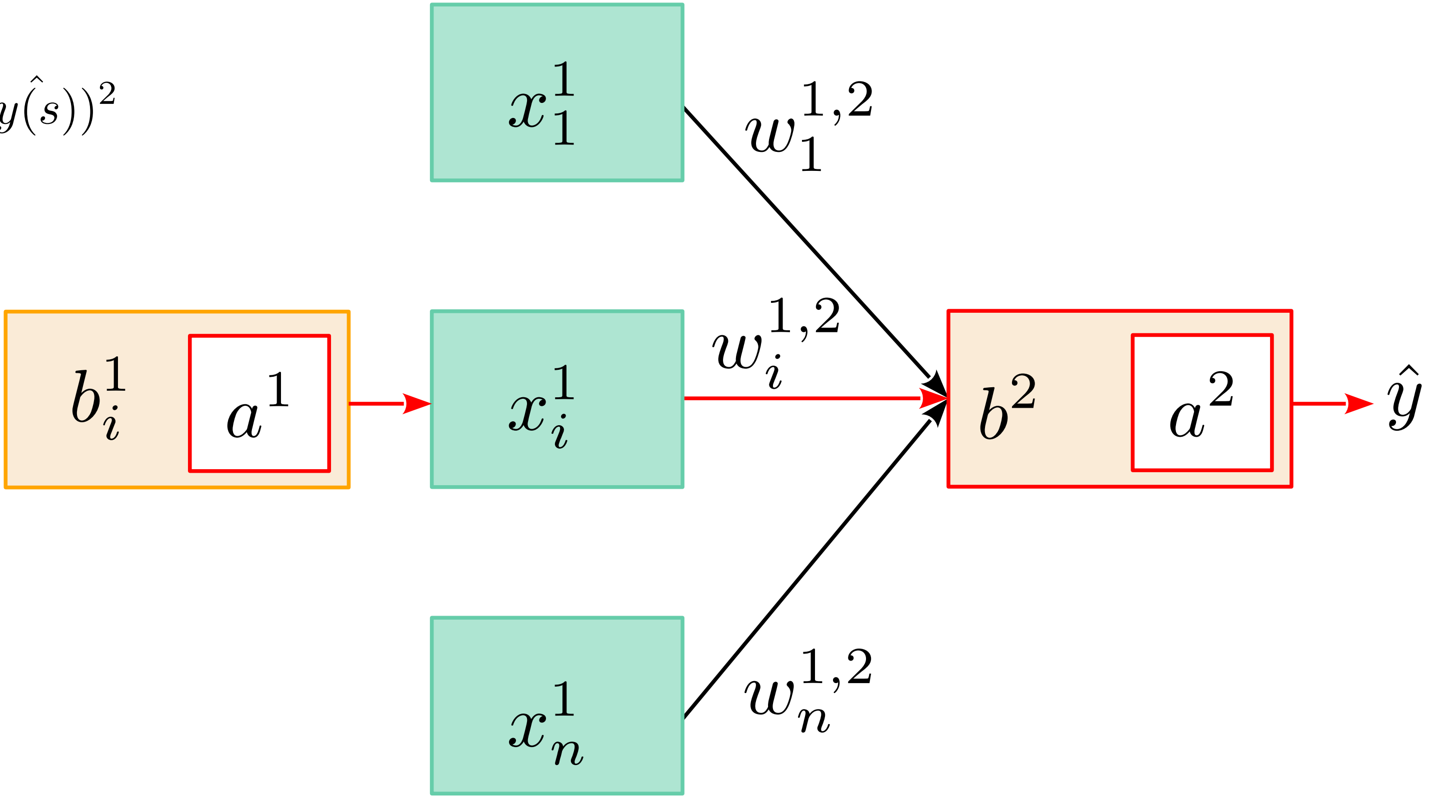
$$\mathrm{MSE} = \frac{1}{S}\sum_{s=1}^{S}(y(s) - y\hat{(s)})^2$$

$$\hat{y} = a^2(z^2)$$

$$z^2 = \sum_{i}^{n} w_i^{1,2} x_i^1 + b^2$$

$$x_i^1 = a^1(z_i^1)$$

...



# Backpropagation: the chain rule

$$h(x) = f(g(x)) \Rightarrow h'(x) = f'(g(x))g'(x)$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \quad \Rightarrow \quad h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \quad \rightarrow \quad h'(x) = f'(g(x))g'(x)$$

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b^2$

$$\frac{\partial \text{MSE}}{\partial b^2} = \frac{\partial \text{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^2} \cdot \frac{\partial z^2}{\partial b^2}$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \quad \longrightarrow \quad h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b^2$

$$\frac{\partial \mathrm{MSE}}{\partial b^2} = \frac{\partial \mathrm{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^2} \cdot \frac{\partial z^2}{\partial b^2}$$

$$\frac{\partial \mathrm{MSE}}{\partial \hat{y}} = -\frac{2}{S} \sum_{s=1}^{S} (y - \hat{y})$$

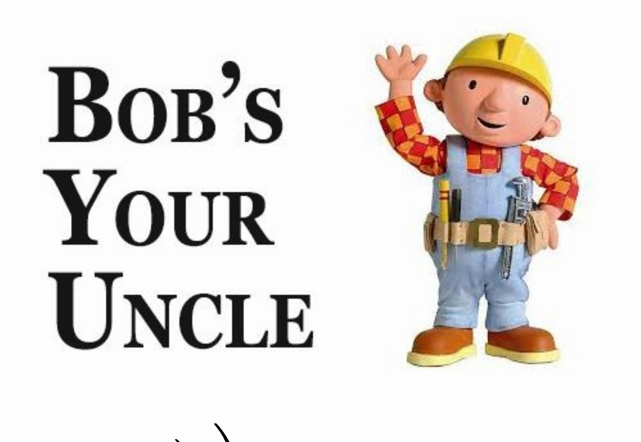
$$\frac{\partial \hat{y}}{\partial z^2} = \frac{1}{1 + e^{-z^2}} \left( 1 - \frac{1}{1 + e^{-z^2}} \right)$$

Recorded forward

$$\frac{\partial z^2}{\partial b^2} = 1$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \longrightarrow h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$




Correction of the bias $b^2$

$$\frac{\partial \text{MSE}}{\partial b^2} = \frac{\partial \text{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^2} \cdot \frac{\partial z^2}{\partial b^2}$$

$$b^2_{\text{new}} = b^2_{\text{old}} - \alpha \left( -\frac{2}{S} \sum_{s=1}^{S} (y - \hat{y}) \cdot \frac{1}{1 + e^{-z^2}} \left( 1 - \frac{1}{1 + e^{-z^2}} \right) \right) \cdot 1$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \quad \longrightarrow \quad h'(x) = f'(g(x))g'(x)$$

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b^2$

$$\frac{\partial \mathrm{MSE}}{\partial b^2} = \frac{\partial \mathrm{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^2} \cdot \frac{\partial z^2}{\partial b^2}$$

$$\downarrow$$

$$\frac{\partial \mathrm{MSE}}{\partial a^2}$$

# Backpropagation: the chain rule

$$h(x) = f(g(x)) \Rightarrow h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b^2$

$$\frac{\partial \mathrm{MSE}}{\partial b^2} = \frac{\partial \mathrm{MSE}}{\partial a^2} \cdot \frac{\partial \hat{y}}{\partial z^2} \cdot \frac{\partial z^2}{\partial b^2}$$

Contribution of $b^2$
to the error of $a^2$'s output

# Backpropagation: feeding errors backwards

$$h(x) = f(g(x)) \quad \Rightarrow \quad h'(x) = f'(g(x))g'(x)$$

$$\frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b_i^1$

$$\frac{\partial \mathrm{MSE}}{\partial b_i^1} = \boxed{\frac{\partial \mathrm{MSE}}{\partial a^2}} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a_i^1} \cdot \frac{\partial a_i^1}{\partial b_i^1}$$

Already computed

# Backpropagation: feeding errors backwards

$$h(x) = f(g(x)) \quad \Rightarrow \quad h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b_i^1$

$$\frac{\partial \mathrm{MSE}}{\partial b_i^1} = \frac{\partial \mathrm{MSE}}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a_i^1} \cdot \frac{\partial a_i^1}{\partial b_i^1}$$

Correction of the weight $w_{1,1}^{0,1}$

$$\frac{\partial \mathrm{MSE}}{\partial w_{1,1}^{0,1}} = \boxed{\frac{\partial \mathrm{MSE}}{\partial a_1^1}} \cdot \frac{\partial a_1^1}{\partial w_{1,1}^{0,1}}$$

Already computed

# Backpropagation: feeding errors backwards

$$h(x) = f(g(x)) \quad \Rightarrow \quad h'(x) = f'(g(x))g'(x) \qquad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Correction of the bias $b_i^1$

$$\frac{\partial \mathrm{MSE}}{\partial b_i^1} = \frac{\partial \mathrm{MSE}}{\partial a^2} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial z^2}{\partial a_i^1} \cdot \frac{\partial a_i^1}{\partial b_i^1}$$

Correction of the weight $w_{1,1}^{0,1}$

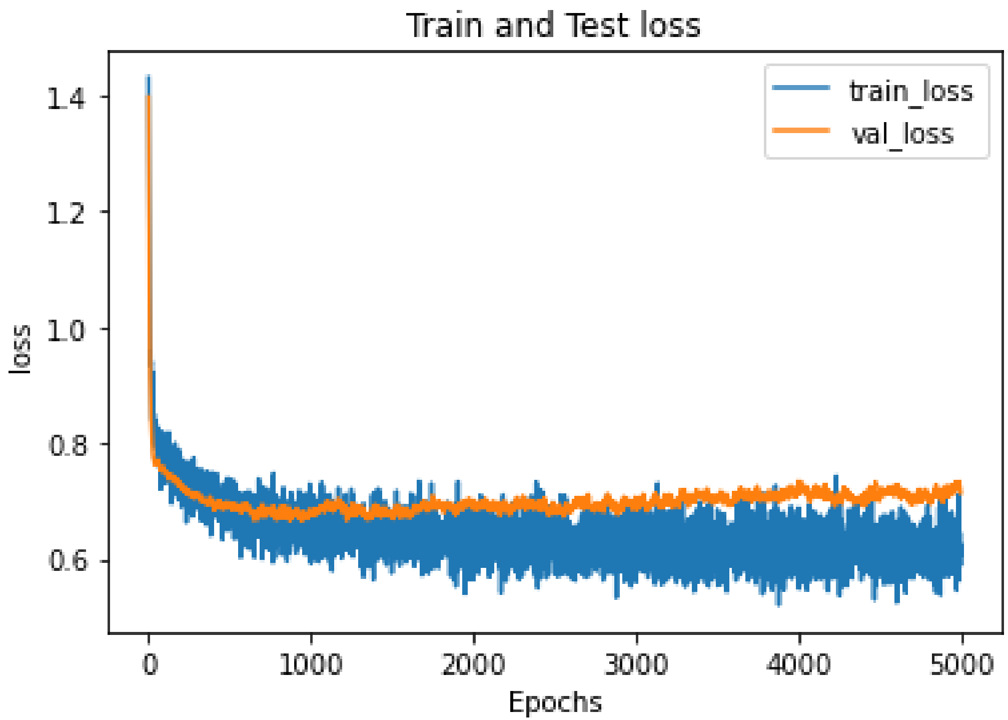
$$\frac{\partial \mathrm{MSE}}{\partial w_{1,1}^{0,1}} = \boxed{\frac{\partial \mathrm{MSE}}{\partial a_1^1}} \cdot \frac{\partial a_1^1}{\partial w_{1,1}^{0,1}}$$

Already computed

All that is done simultaneously for all weights and biases of a layer using tensors and Hadamard products (i.e. element-wise multiplication $\odot$ )

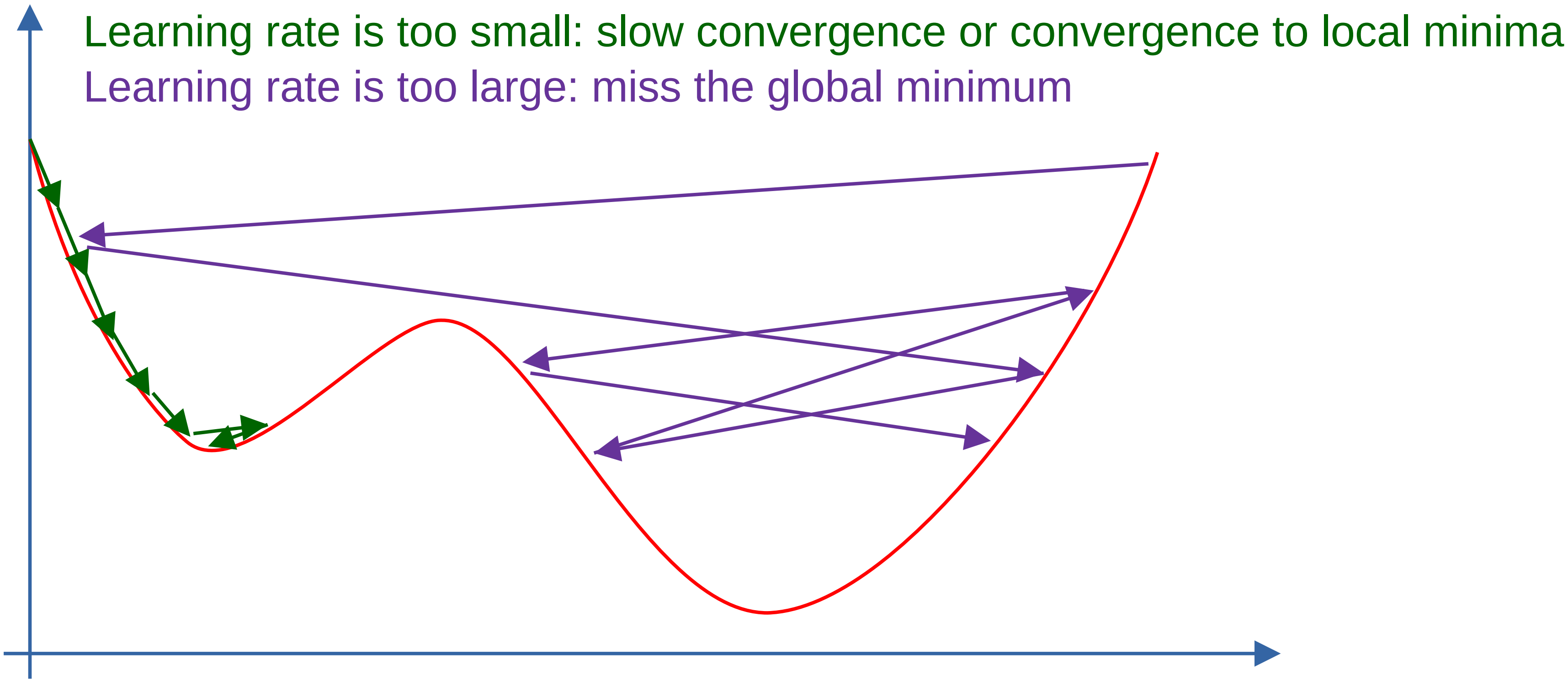
To know much more than you would like:
http://neuralnetworksanddeeplearning.com/chap2.html and
LeCun, Bengio, and Hinton (2015) Deep learning. *Nature*, 521 (7553), pp.436-444.

# Learning



Train and Test loss
train_loss
val_loss
loss
1.4
1.2
1.0
0.8
0.6
0
1000
2000
3000
4000
5000
Epochs

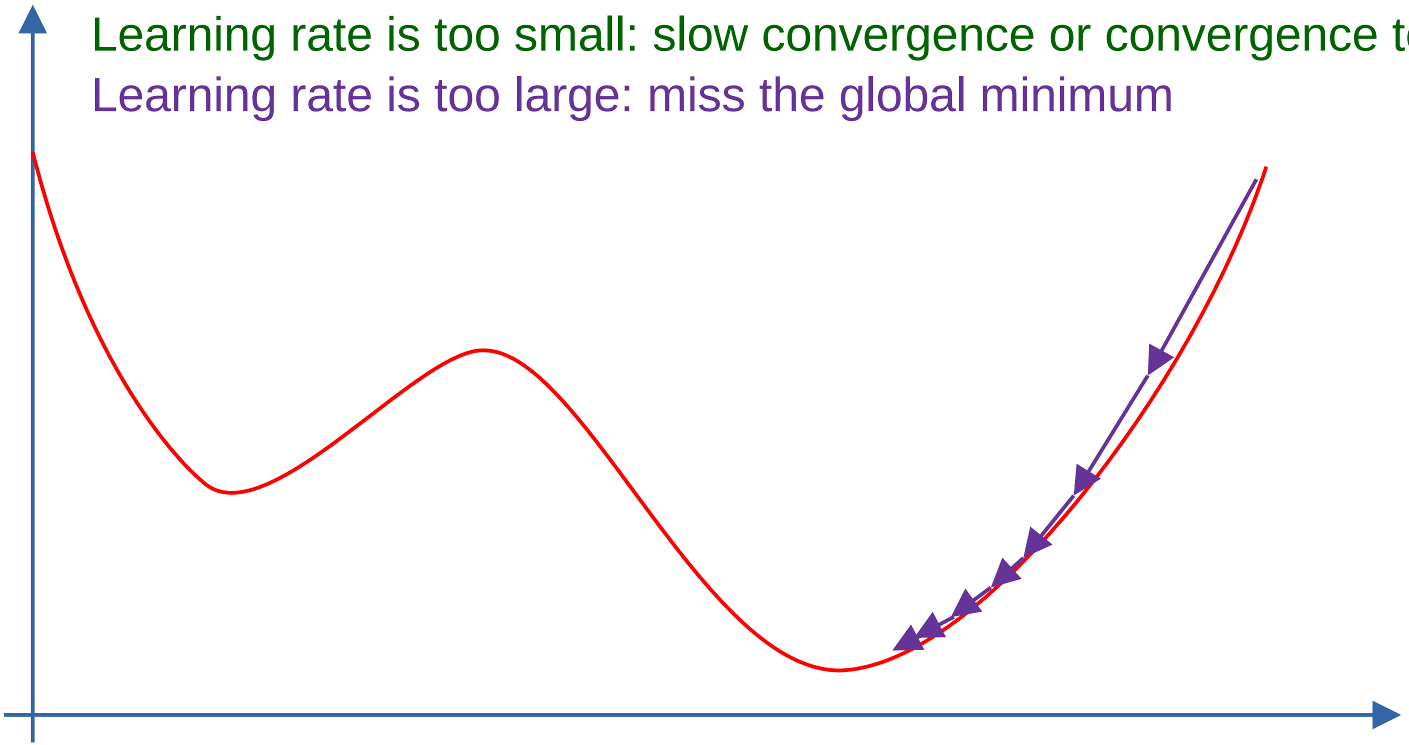
# Impact of the learning rate



# Impact of the learning rate

Learning rate is too small: slow convergence or convergence to local minima

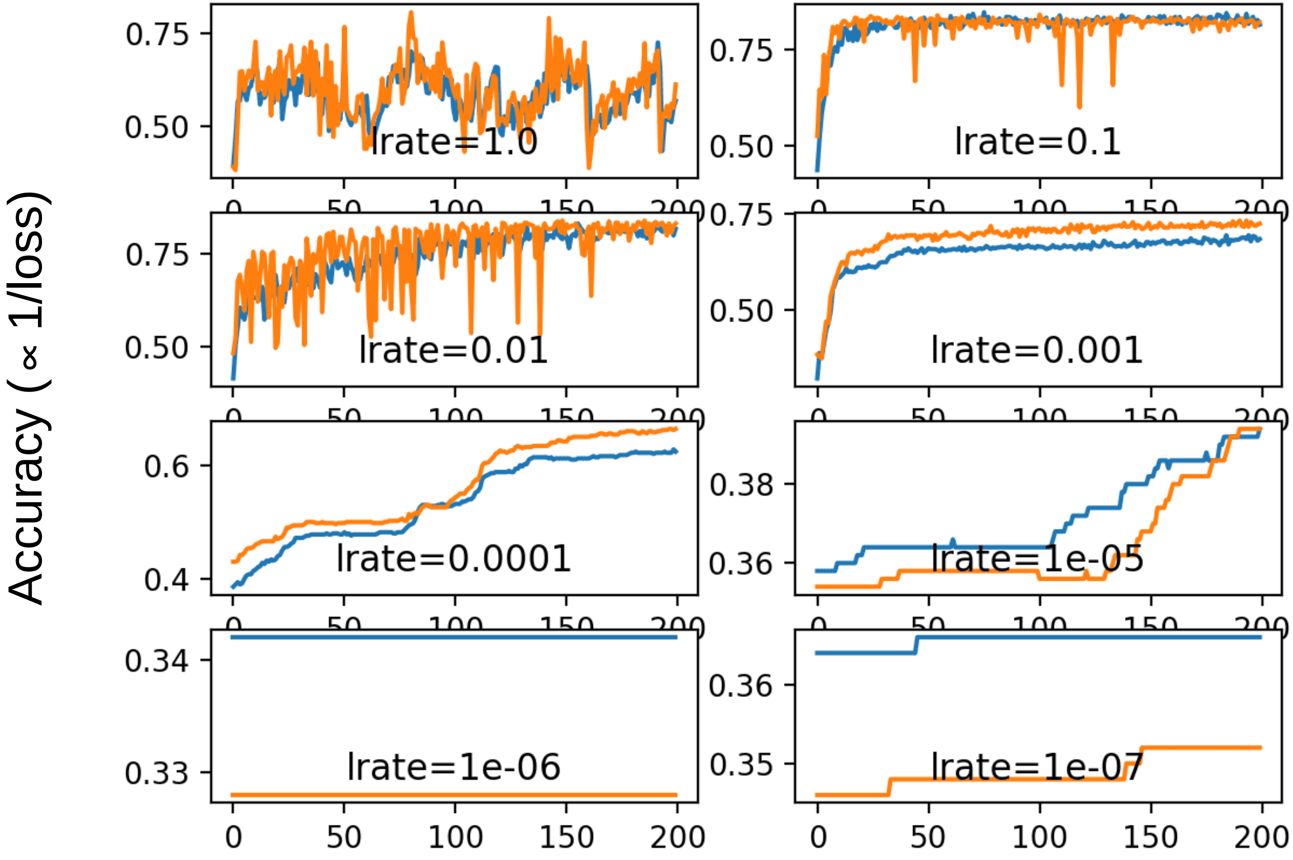
Learning rate is too large: miss the global minimum



Adaptive algorithms adapt the learning steps according to the gradient of the cost function

Kingma and Lei Ba (2015) Adam: a method for stochastic optimisation. *ICLR 2015*

# Impact of the learning rate



# Training, testing, and validation sets

“validation” (never seen)
Same for all model instances
Used to assess the model at the end

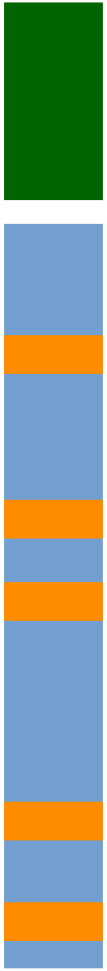
Training set: used to learn

(perso: training + test set = learning set)

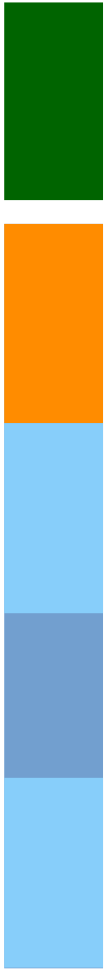
“test” set: used to assess the model
during the learning phase
Different for each model instance

**<u>Beware</u>**: “validation” and “test” are used the other way around a lot in deep learning, at the opposite of all other fields of machine learning, or even life science in general

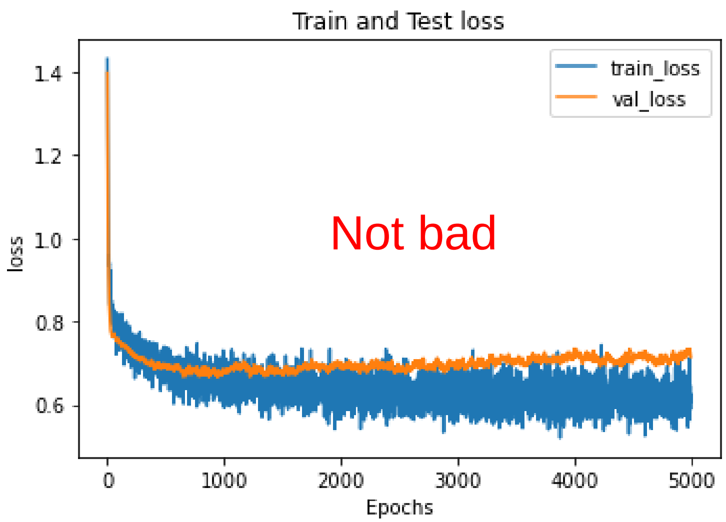
Random
Test samples



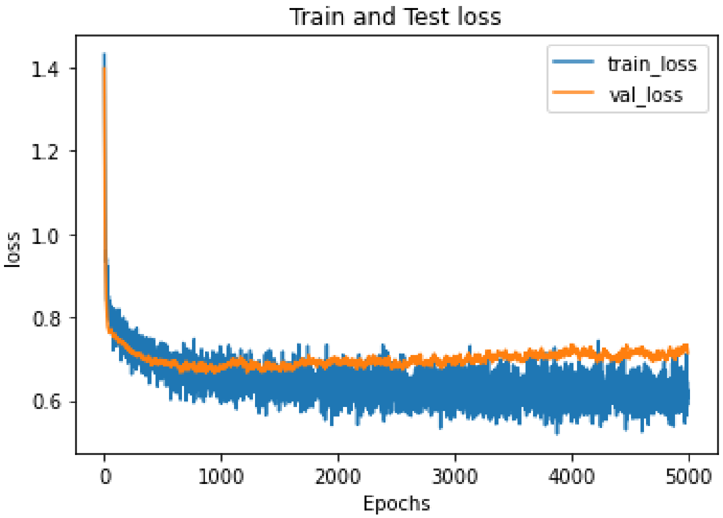
K-fold
validation

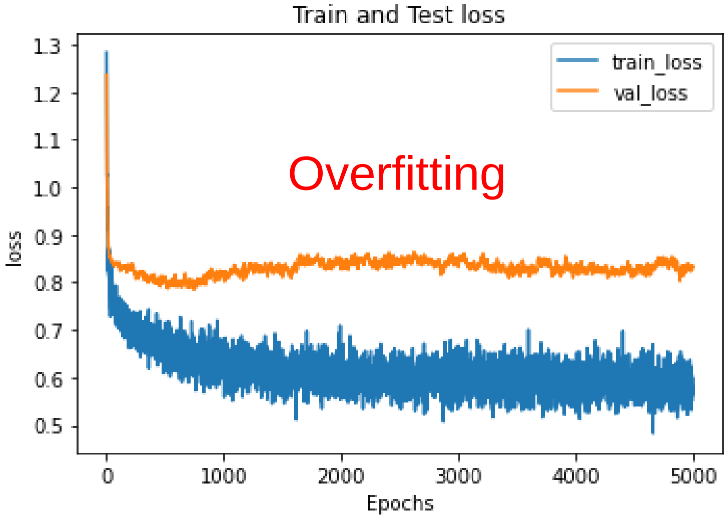


# Learning and overfitting

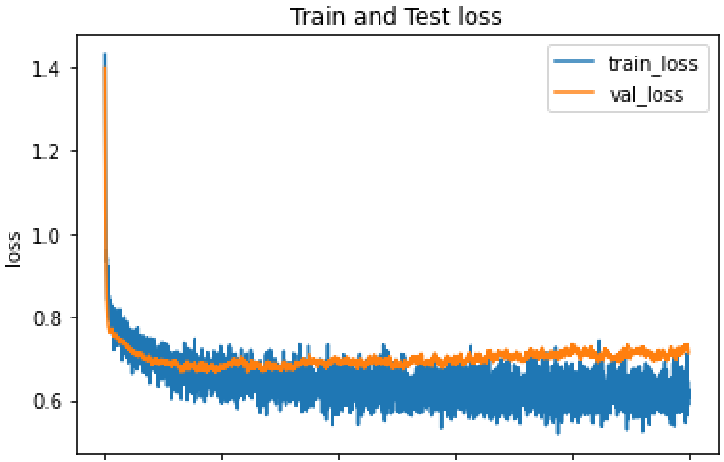


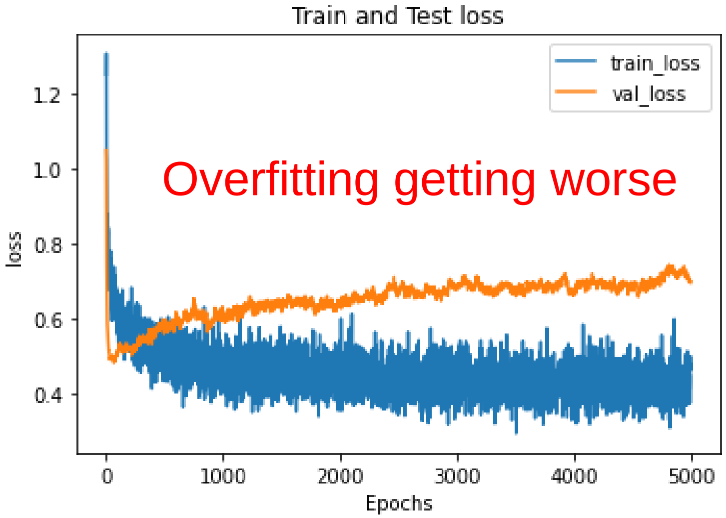
# Learning and overfitting

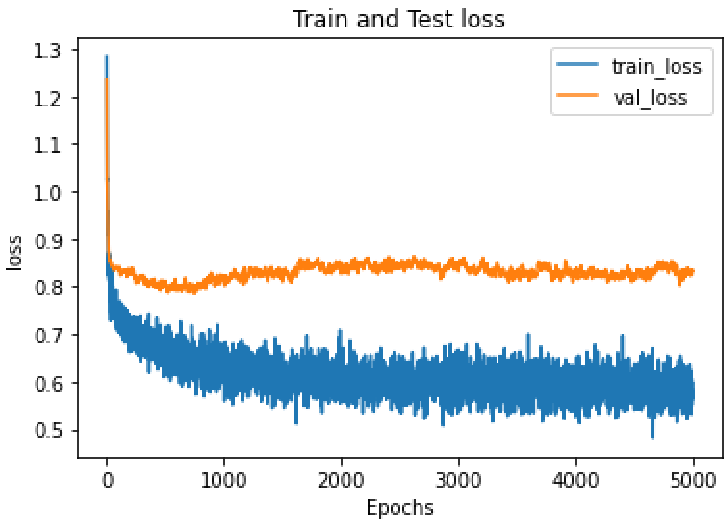


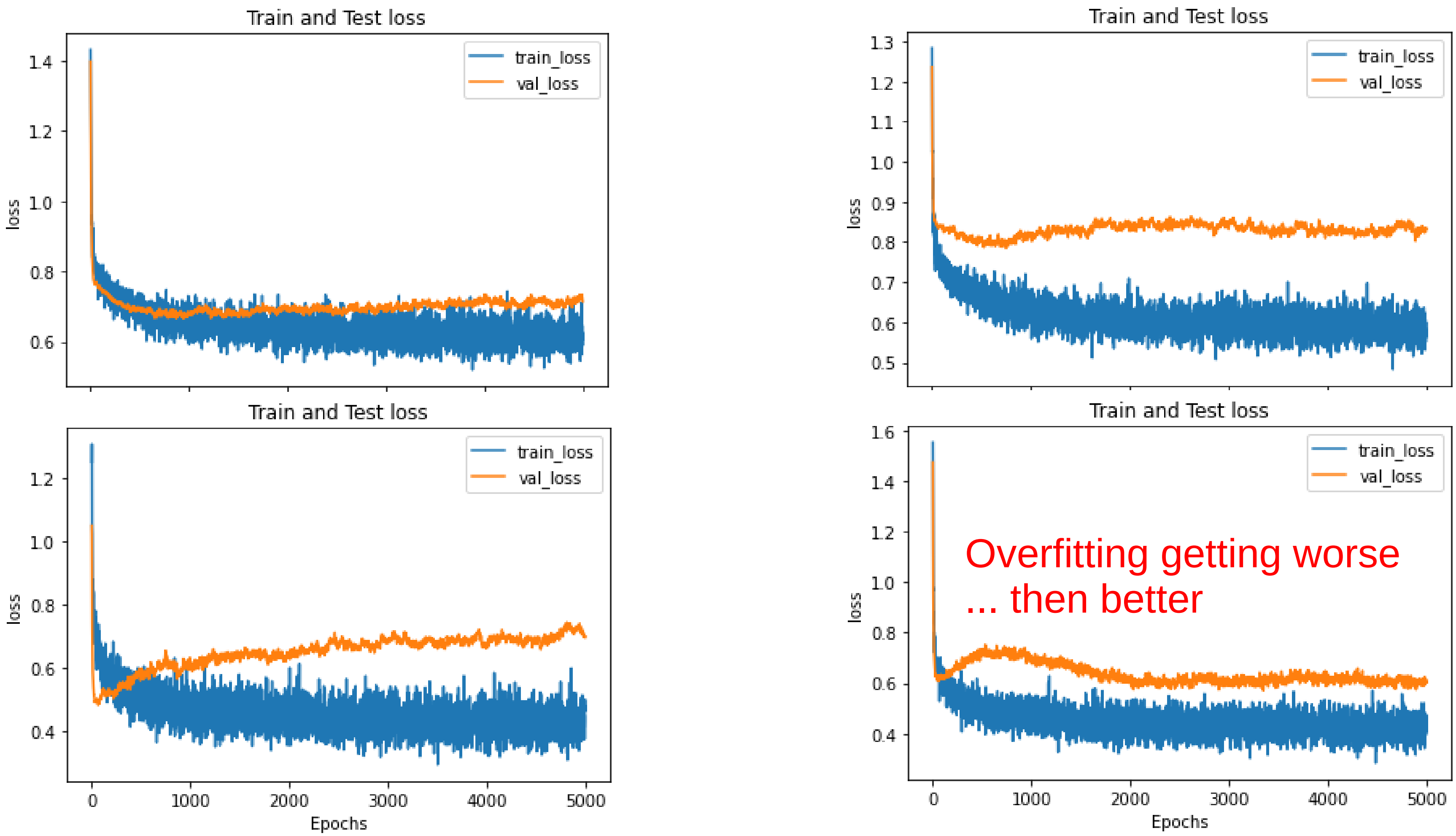
# Learning and overfitting



# Learning and overfitting



# Why a test AND a validation set?

Underperforming on the test set means the model overfitted the training set, the parameters are too specific of the samples in the training set

Underperforming on the validation set means the hyperparameters are too specific of the test set as well! When you modify the structure of the model to avoid overfitting, you actually make the model overfit the entire learning set

# Let’s predict the diabetes status

Dataset: Pima Indians Diabetes Database. Smith *et al* (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press

Variables:

1) Number of times pregnant
2) Plasma [glucose] at 2 h in an OGTT
3) Diastolic blood pressure (mm Hg)
4) Triceps skin fold thickness (mm)
5) 2-Hour serum insulin (mu U/ml)
6) Body mass index (weight in kg/(height in m)^2)
7) Diabetes pedigree function
8) Age (years)

   +

9) Diabetes status: 500 healthy (0), 268 diabetic (1)

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 2 | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 3 | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 4 | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 5 | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |
| 6 | 5 | 116 | 74 | 0 | 0 | 25.6 | 0.201 | 30 | 0 |
| 7 | 3 | 78 | 50 | 32 | 88 | 31 | 0.248 | 26 | 1 |
| 8 | 10 | 115 | 0 | 0 | 0 | 35.3 | 0.134 | 29 | 0 |
| 9 | 2 | 197 | 70 | 45 | 543 | 30.5 | 0.158 | 53 | 1 |
| 10 | 8 | 125 | 96 | 0 | 0 | 0 | 0.232 | 54 | 1 |
| 11 | 4 | 110 | 92 | 0 | 0 | 37.6 | 0.191 | 30 | 0 |
| 12 | 10 | 168 | 74 | 0 | 0 | 38 | 0.537 | 34 | 1 |
| 13 | 10 | 139 | 80 | 0 | 0 | 27.1 | 1.441 | 57 | 0 |
| 14 | 1 | 189 | 60 | 23 | 846 | 30.1 | 0.398 | 59 | 1 |

# Software stack

| | |
|---|---|
| High-level library | Keras |
| Low-level library | TensorFlow |
| Language | python™ |
| IDE | colab, spyder |

# Software stack

| | |
|---|---|
| High-level library | Keras |
| Low-level library | TensorFlow |
| Language | python™ |
| IDE | colab, spyder |

But instead
you c/should use

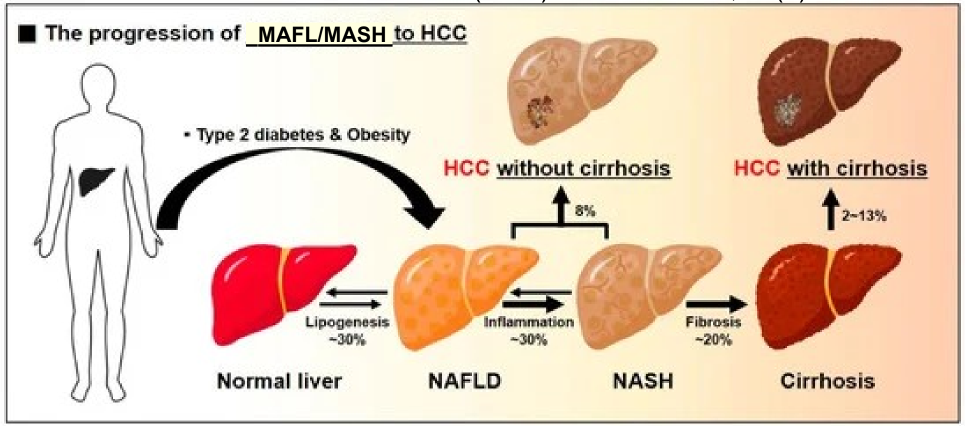
PyTorch

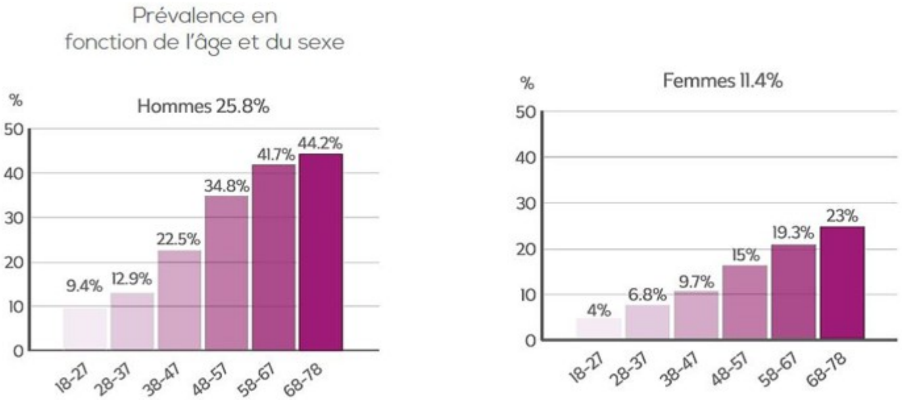
jupyter

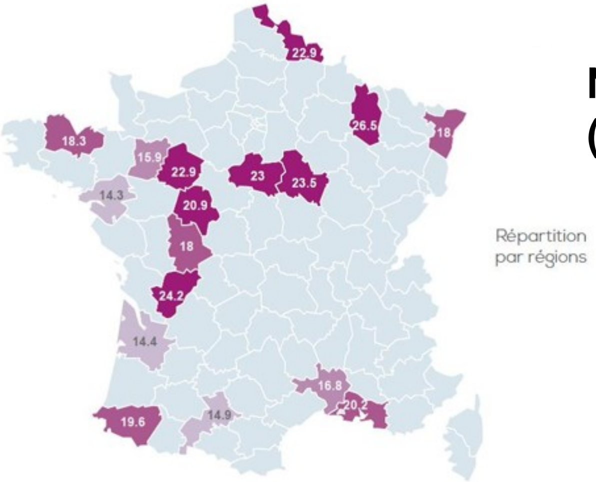
# Demo in Google Colab

# Let’s try to recognise a disease severity

Kim *et al* (2021) *Int. J. Mol. Sci.*, 22(9): 4495

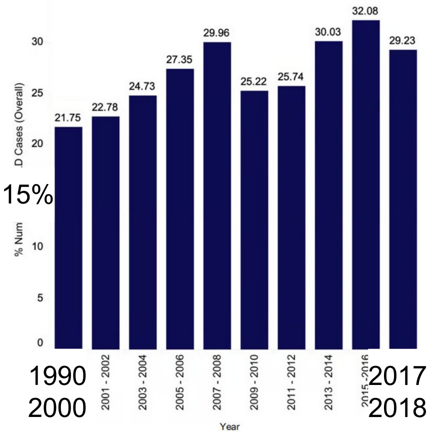


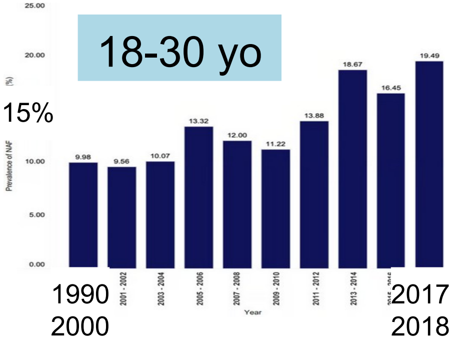




**NASH (now MASH)**

Paris MASH Meeting (11-12 juillet 2019)



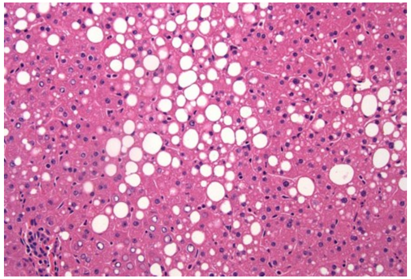

Kim *et al* (2022) *Met. Target Organ Damage*, 2: 19

**NALFD (now MASLD)**

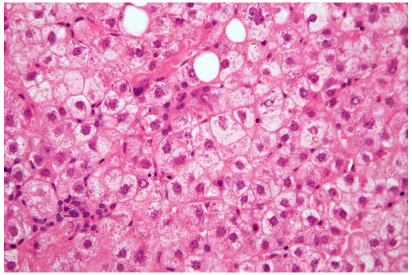
# Subject grouping

Scoring on liver biopsy with the method from Kleiner and Brunt 2005

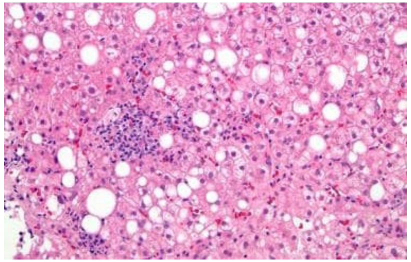
**Steatosis**
Categorical [0-3] from
quantitative measurement



**Ballooning**
Categorical [0-2]
= {none, some, much}

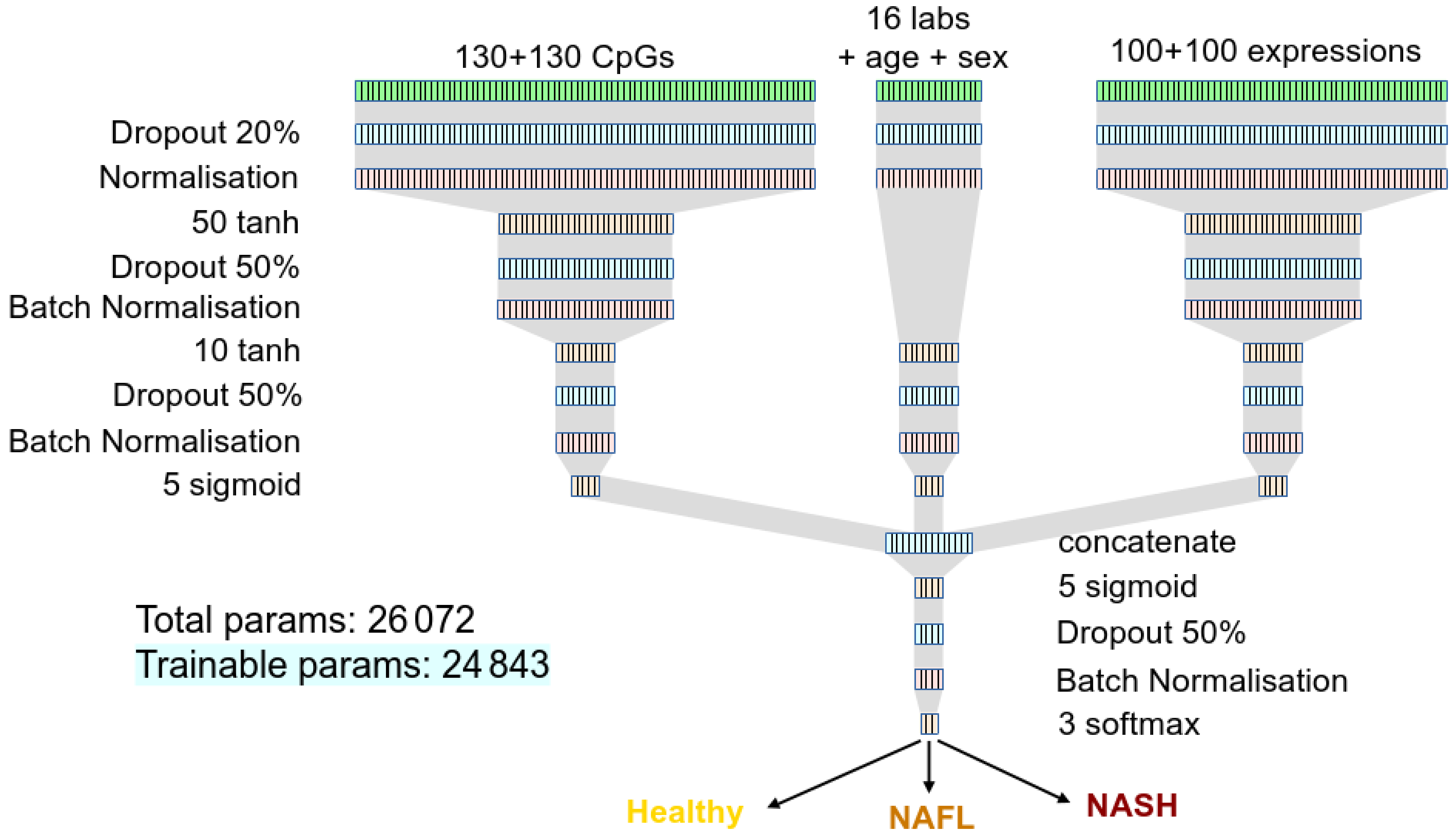


**Inflammation**
Categorical [0-3] from
number of foci



Final score:

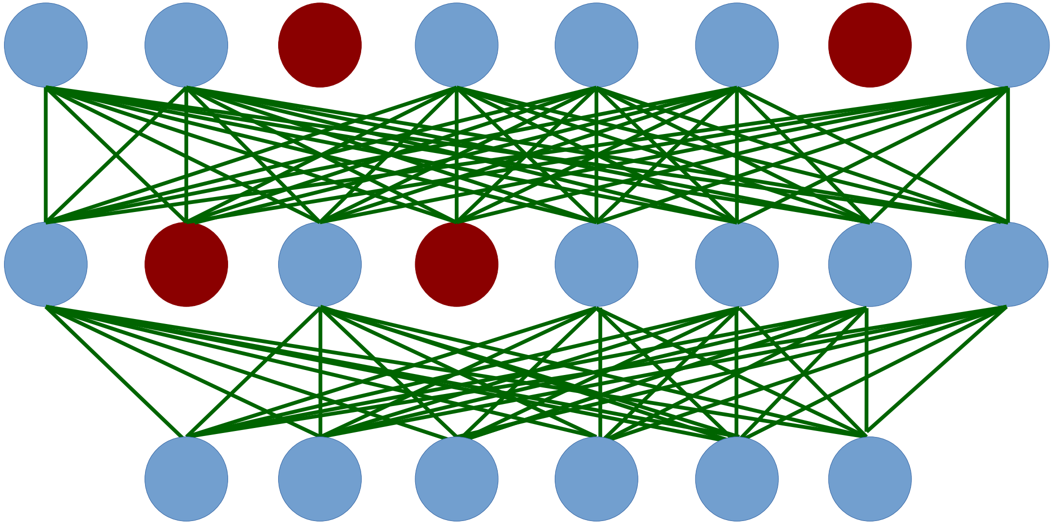
| | | |
|---|---|---|
| **Healthy**: | $S = 0,\ B = 0,\ I = 0$ | $n = 80$ |
| **NAFL**: | $S > 1,\ B = 0,\ I \geq 1$<br>$S > 1,\ B > 1,\ I = 0$ | $n = 137$ |
| **NASH**: | $S > 0,\ B > 0,\ I > 0$ | $n = 83$ |



130+130 CpGs
16 labs
+ age + sex
100+100 expressions
Dropout 20%
Normalisation
50 tanh
Dropout 50%
Batch Normalisation
10 tanh
Dropout 50%
Batch Normalisation
5 sigmoid
concatenate
5 sigmoid
Dropout 50%
Batch Normalisation
3 softmax
Total params: 26 072
Trainable params: 24 843
Healthy
NAFL
NASH

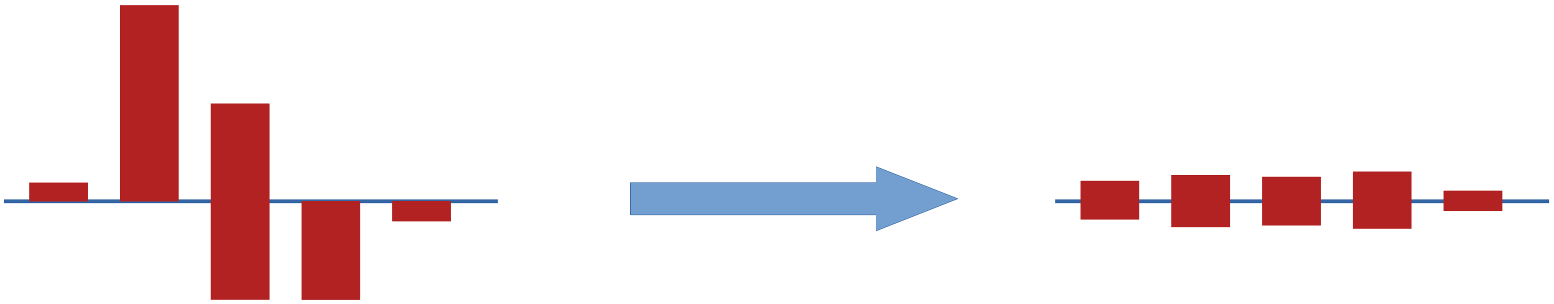
# Dropout

- Purpose: avoiding overfitting
- Disable some connections at random (set the weights at 0)



Srivastava et al (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR 15(56):1929−1958

# Normalisation



- Normalisation during data processing
- Normalisation before training: on the whole dataset
- Normalisation during training: after dropout
- Batch normalisation: normalisation on the current batch (not on the entire dataset)
- Layer normalisation: normalisation of the input of a layer

Ba, Kiros, Hinton (2016) Layer Normalization. arXiv:1607.06450v1
Ioffe and Czegedy (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proc 32nd Intl Conf Machine Learning, Lille, France volume 37

# Balance the datasets!

If your learning dataset is made up of 90% of class A and 10% of class B, you reach 90% accuracy by predicting all cases as A...

Selective sampling:

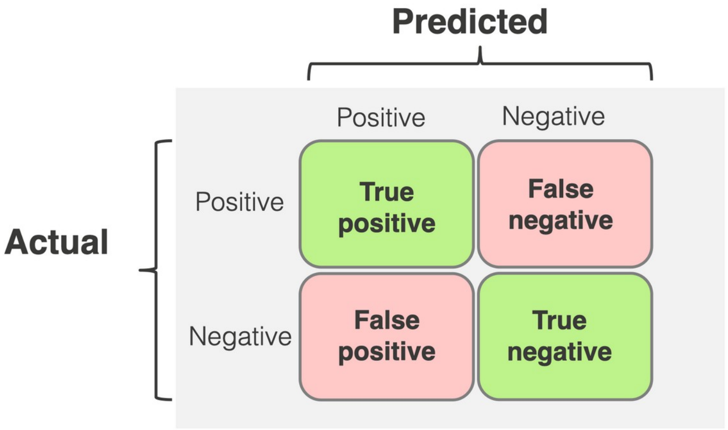
Randomly select the same number of class A as in class B

Data augmentation:

Sample repeatedly in class B
Synthesise class B samples

# Demo in Spyder

# Evaluating a model's performance



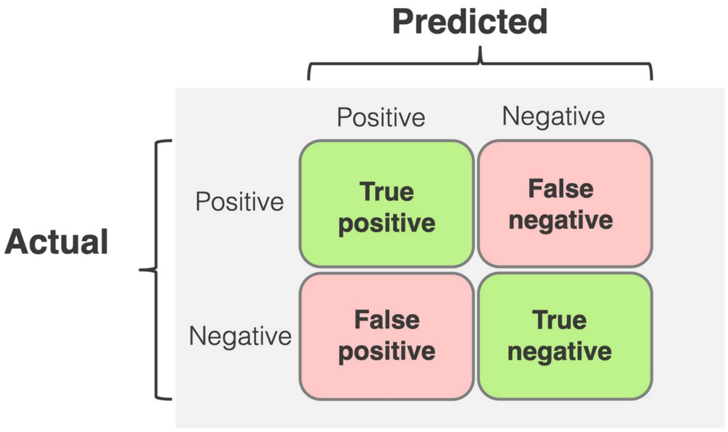

Accuracy = (TP+TN)/(TP+FN+TN+FP)

Precision = TP/(TP+FP)

Sensitivity (true positive rate) = TP/(TP+FN)

Specificity (true negative rate) = TN/(TN+FP)

# Evaluating a model's performance



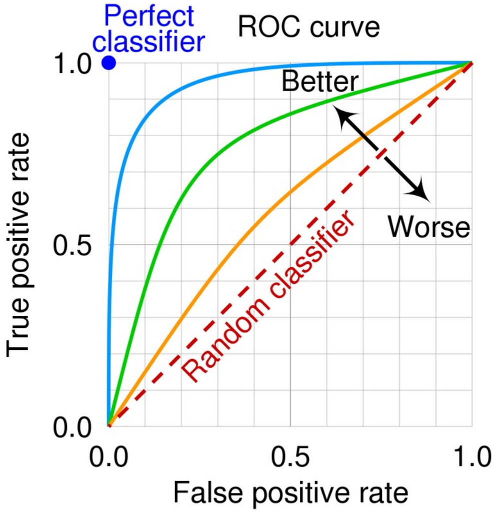

Accuracy = (TP+TN)/(TP+FN+TN+FP)

Precision = TP/(TP+FP)

Sensitivity (true positive rate) = TP/(TP+FN)

Specificity (true negative rate) = TN/(TN+FP)

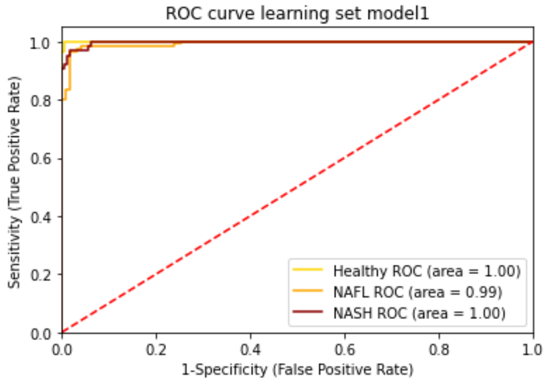
Receiver operating characteristic (ROC) curve

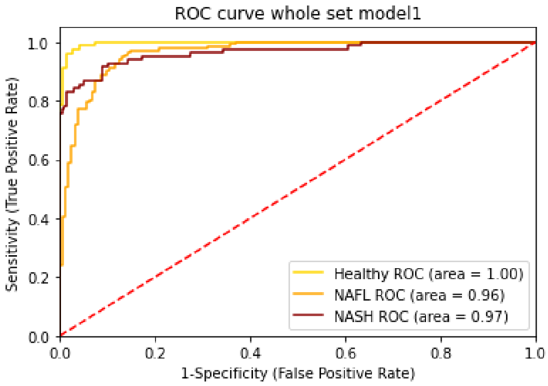


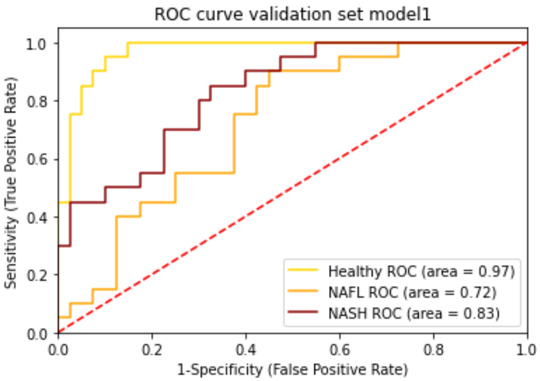
# Different accuracies on different datasets

Learning set | Whole set | Validation set

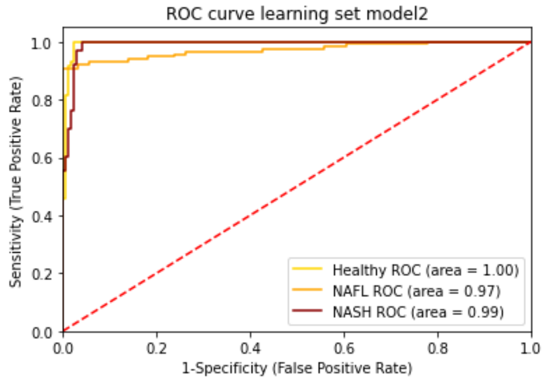
Model 1

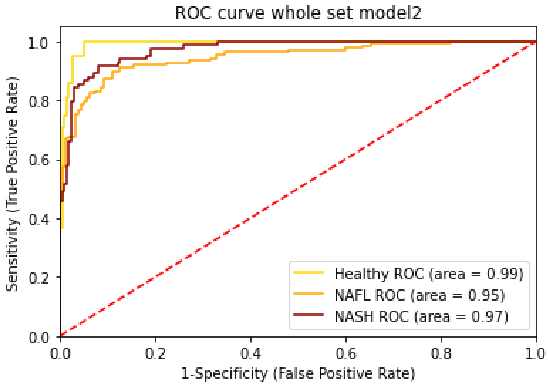


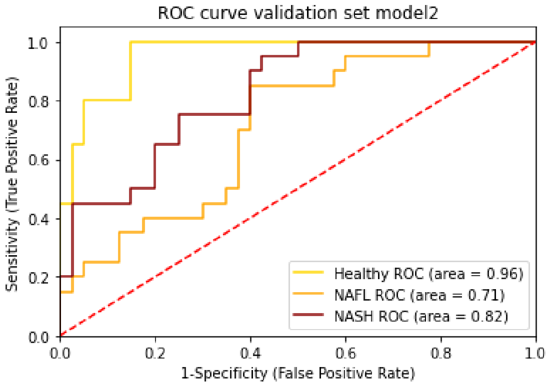




Model 2

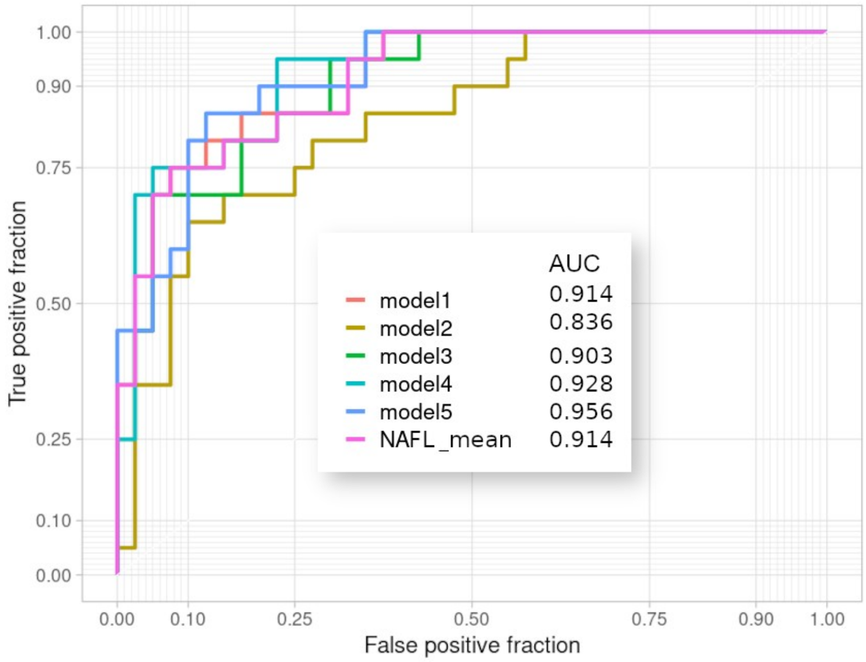


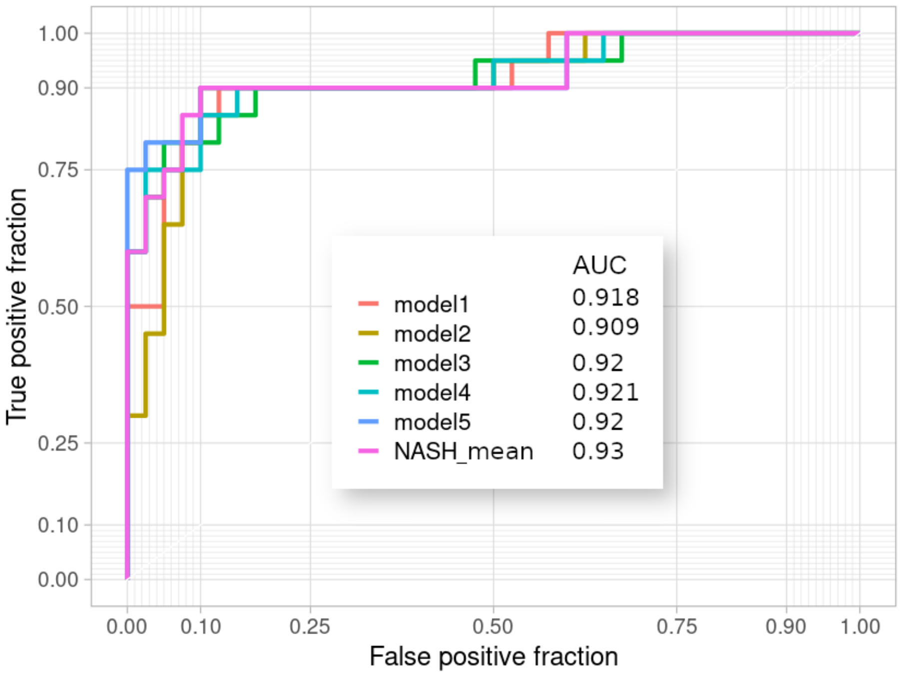



# How good is the model to distinguish NAFL and NASH?

sensitivity



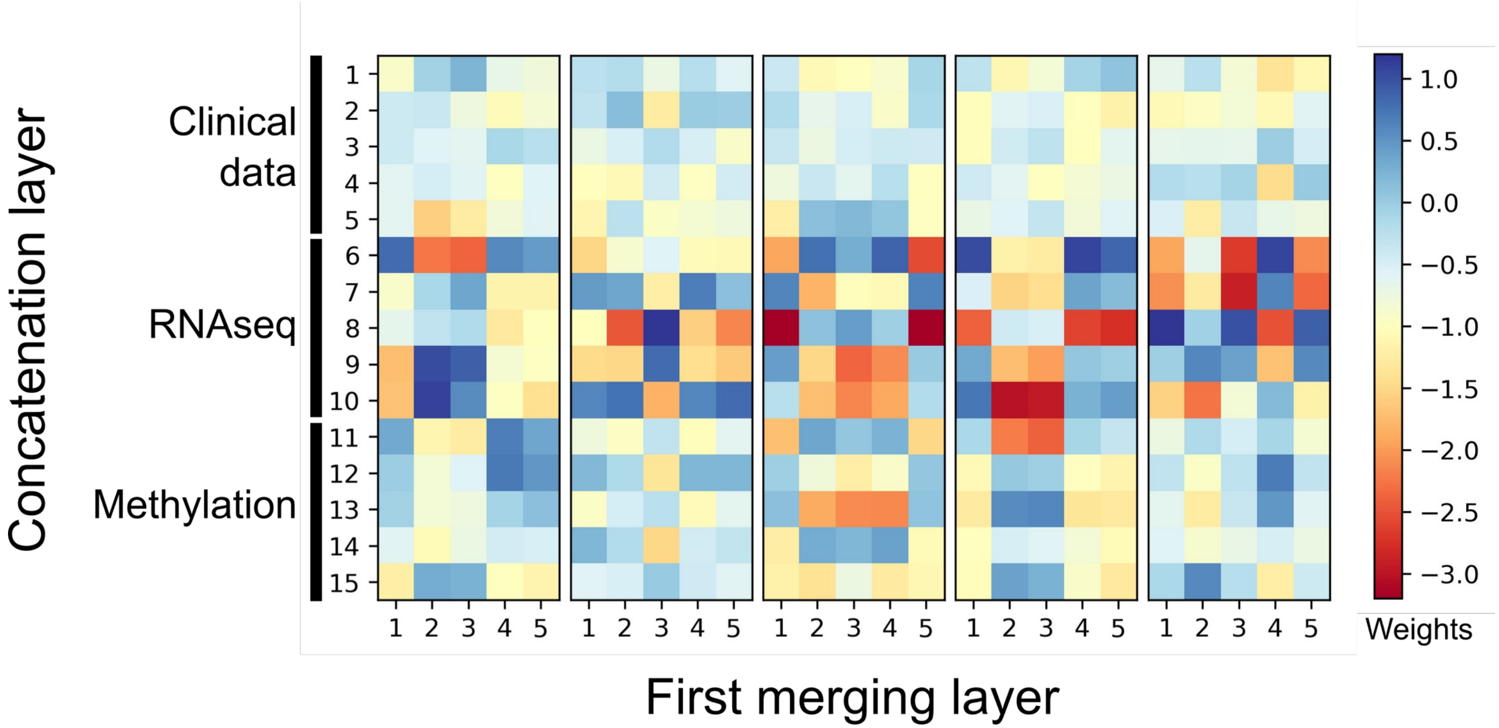

1 - specificity



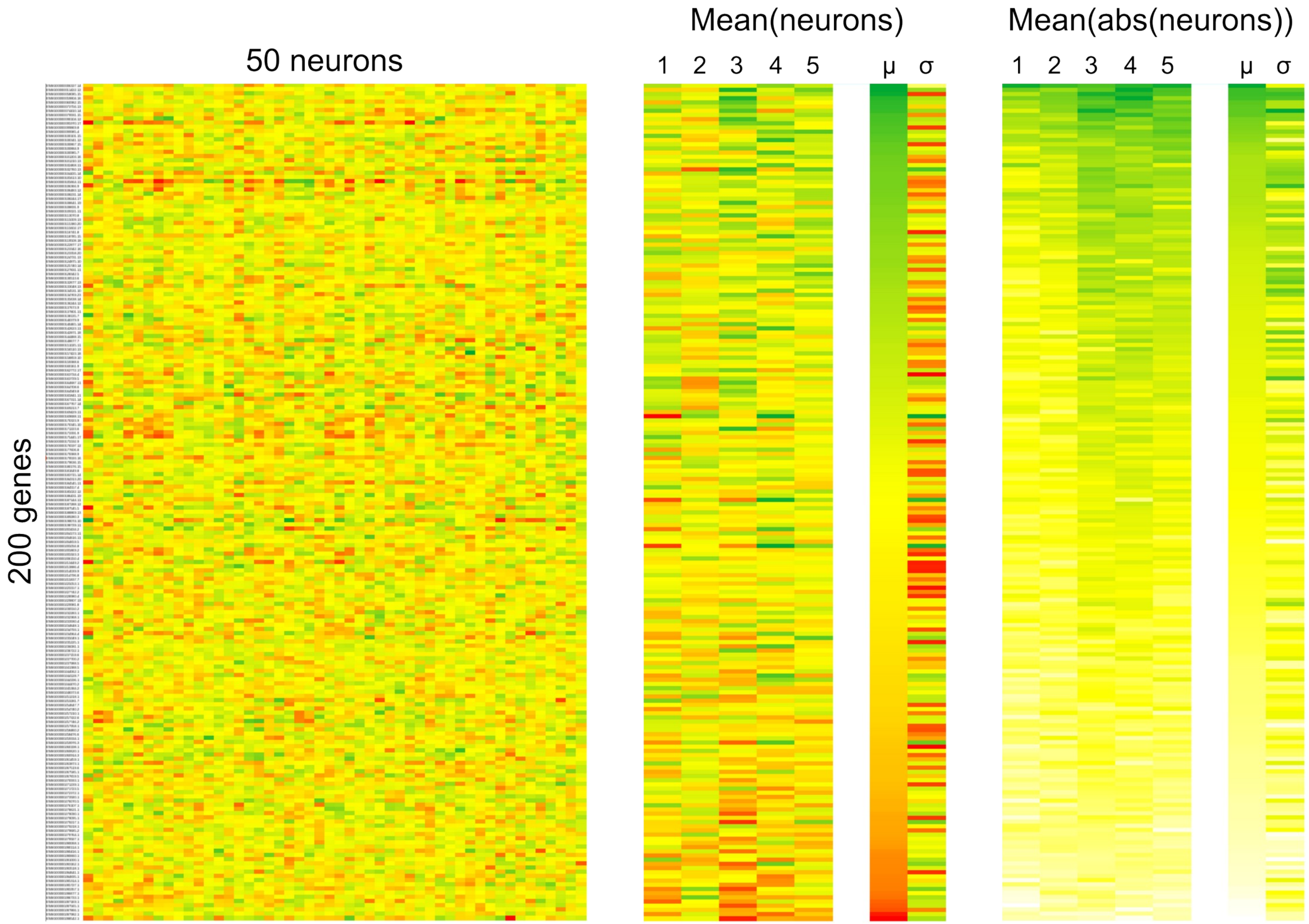

1 - specificity

# Peeping into the black box

The RNAseq module has the most impact on output



# Independently trained models learn from the same genes



# Want to go further?

Fidle:
https://fidle.cnrs.fr/w3/
https://www.youtube.com/@CNRS-FIDLE

StatQuest: Start by
https://www.youtube.com/watch?v=GKZoOHXGcLo
And move backward to get to the start you need.

3 blue 1 brown:
https://www.youtube.com/@3blue1brow
Search for deep learning, chapter 1 to 4

Also, a large number of tutorial with code at:
https://machinelearningmastery.com/category/deep-learning/