Can you recognise the music playing?

6COSC020W: APPLIED AI

WEEK 8: ARTIFICIAL NEURAL NETWORKS (ANNs)

DR. ESTER BONMATI

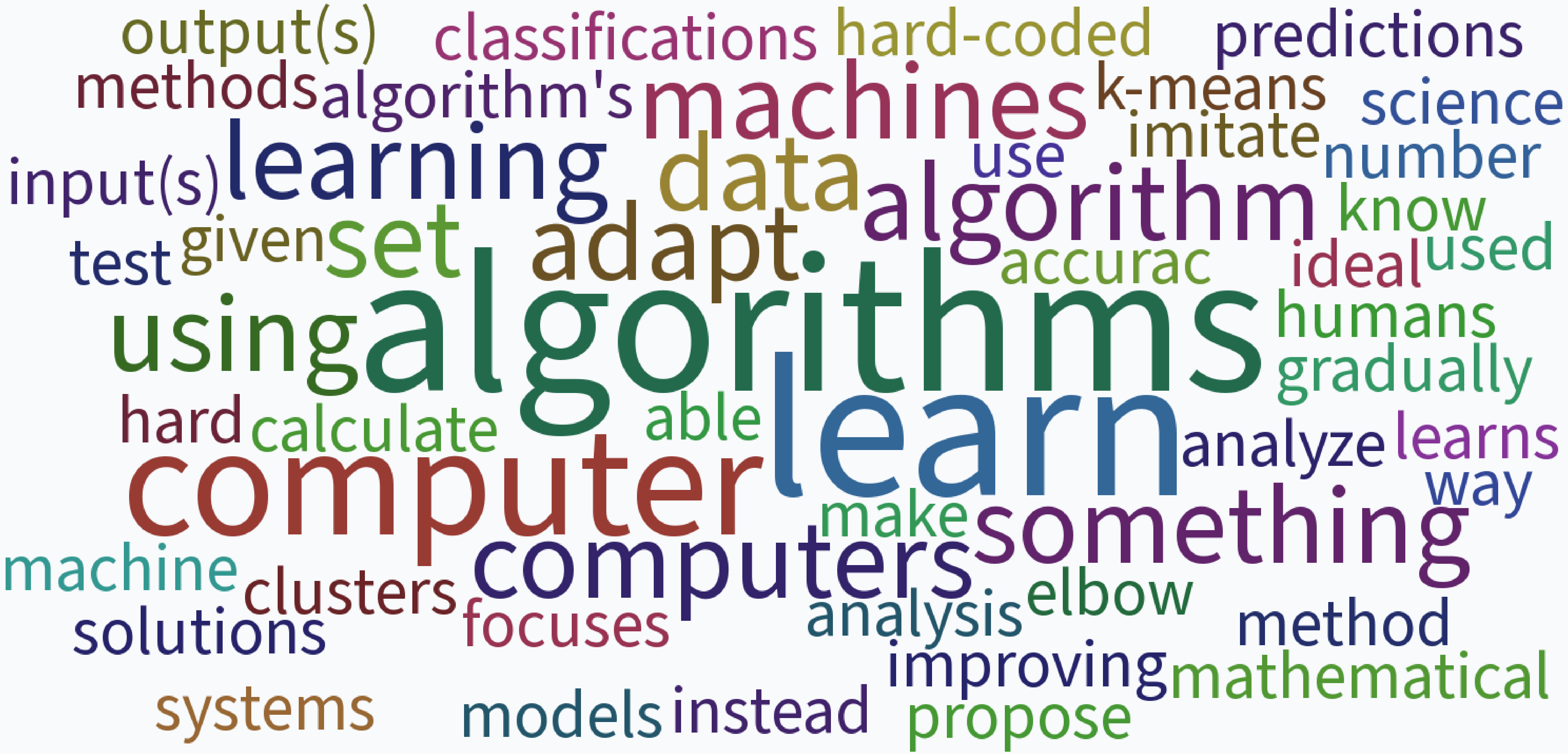
(e.bonmaticoll@westminster.ac.uk)

NOTES

- We will use PollEverywhere multiple times during the lecture: **polleverywhere.com/esterbonmati**
- It may be useful to have a notebook and pen/pencil (or your preferred method) to take some notes.

Before session: What do you think machine learning is?

FROM LAST SESSION...



LEARNING OUTCOMES OF THIS SESSION

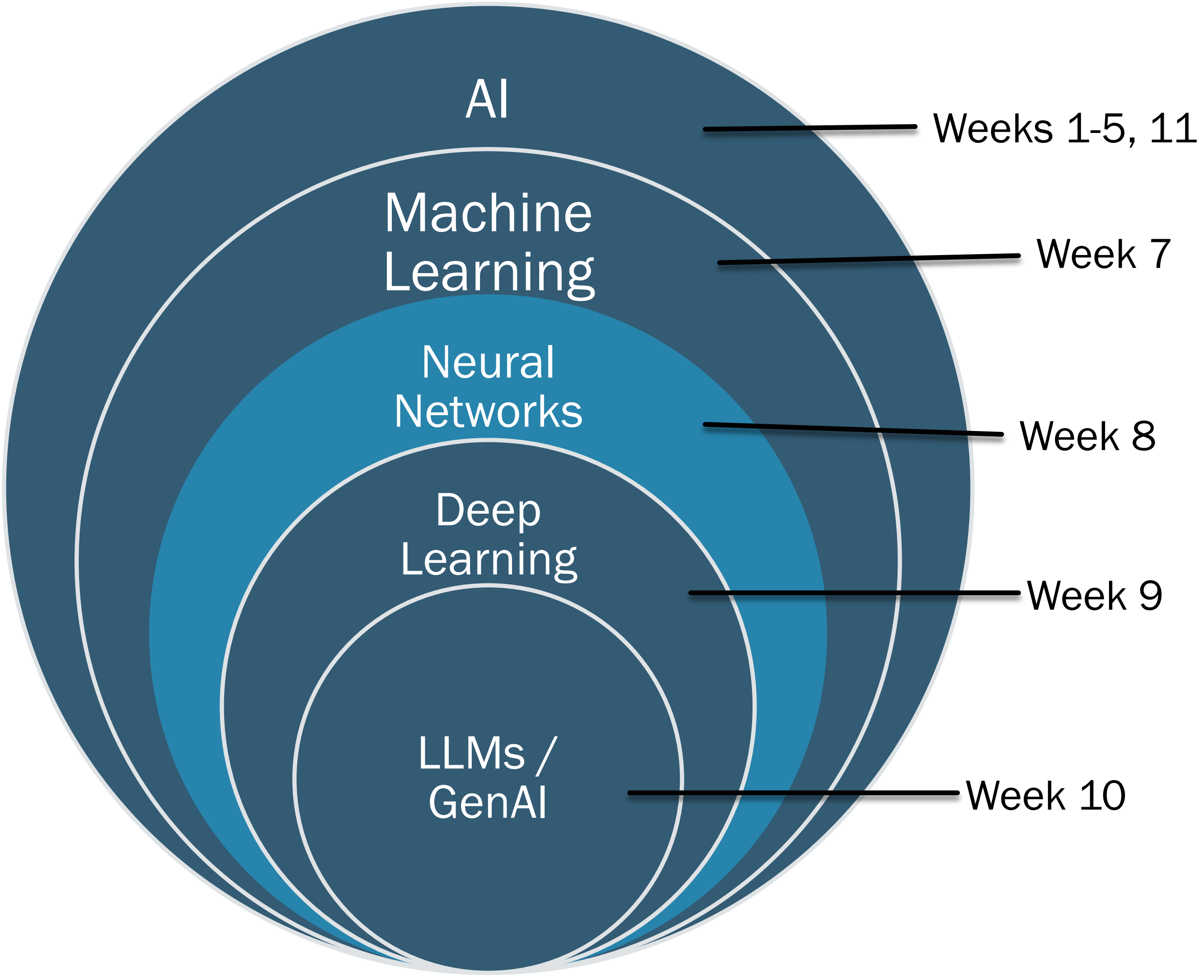
Perceptron

- Explain the perceptron
- Inputs and outputs
- Weights
- **Forward propagation**
- **Explain what is an activation functions**
- **Be able to calculate the output of a perceptron**

Neural Networks

- **Be able to list the name of the different layers**
- **Be able to describe how neural networks work in your own words**
- Explain problems with data
- Explain what are hyperparameters
- Be able to evaluate performance and calculate accuracy

ARTIFICIAL INTELLIGENCE: MACHINE LEARNING



Before: what is an artificial neural network?

pollev.com/esterbonmati

Nobody has responded yet.

Hang tight! Responses are coming in.

Do you know any examples of applications using a neural network?

pollev.com/esterbonmati

Nobody has responded yet.

Hang tight! Responses are coming in.

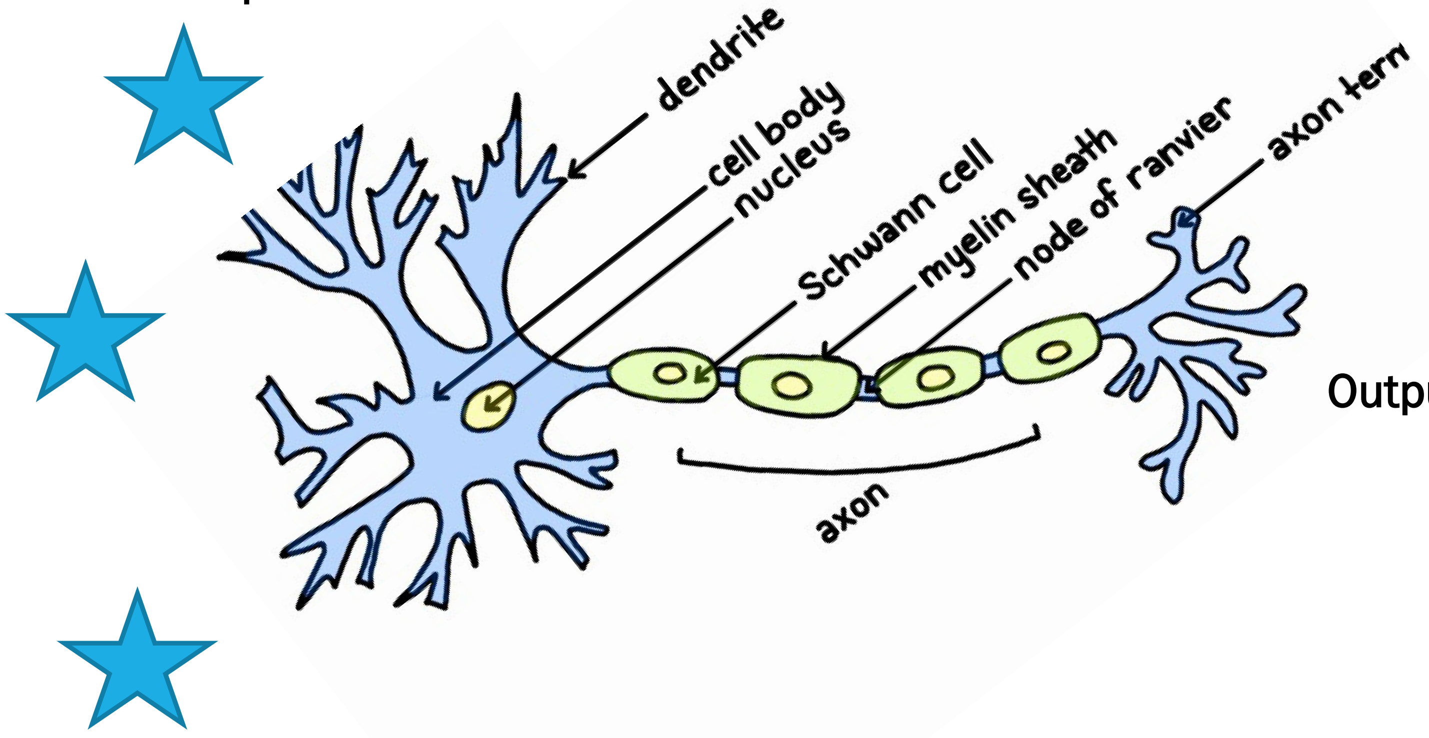
ARTIFICIAL NEURAL NETWORKS (ANNs)

Why are they important?

ANN can **help computers make intelligent decisions with limited human assistance.** They can learn and model the **relationships** between input and output data that are nonlinear and complex.

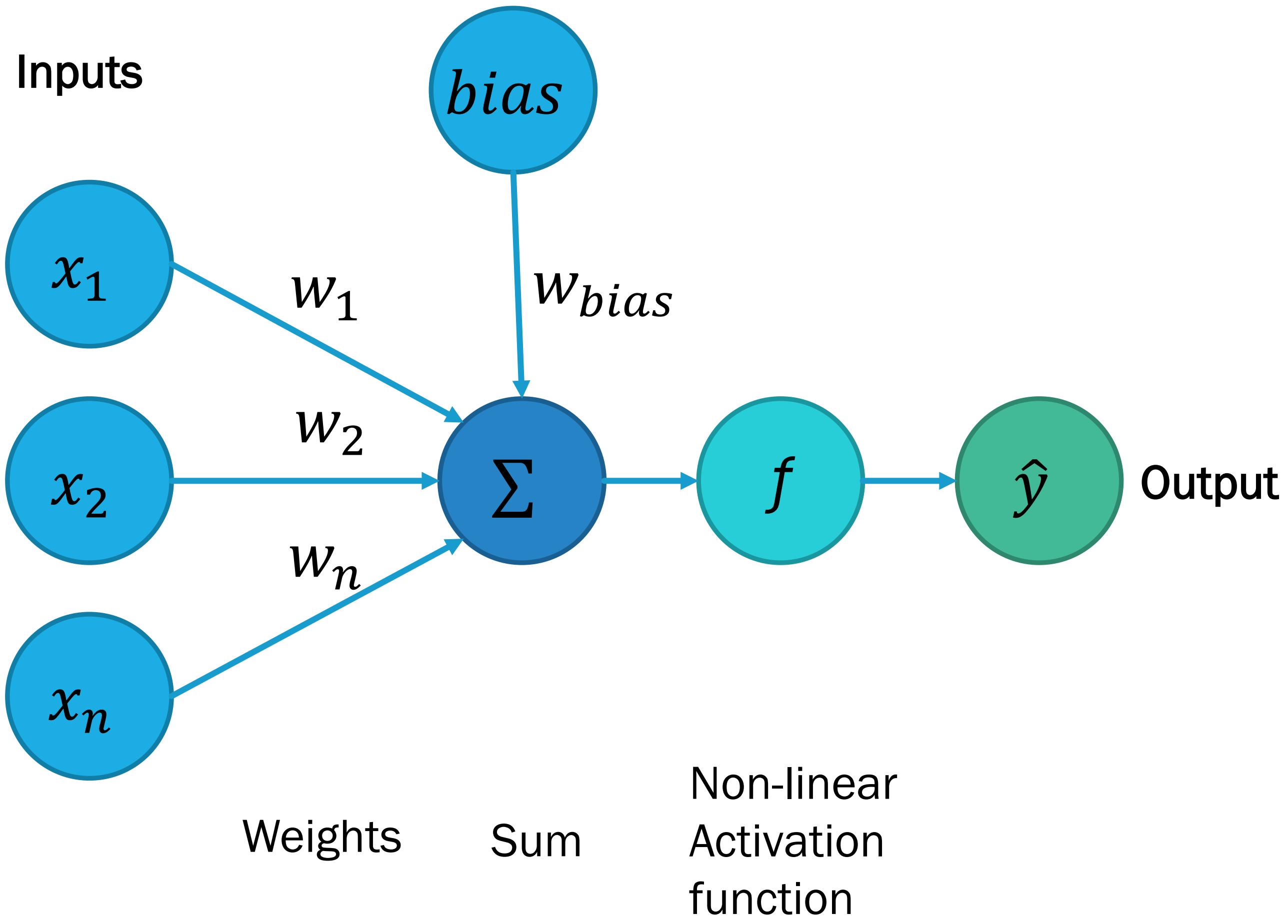
NEURON

Inputs



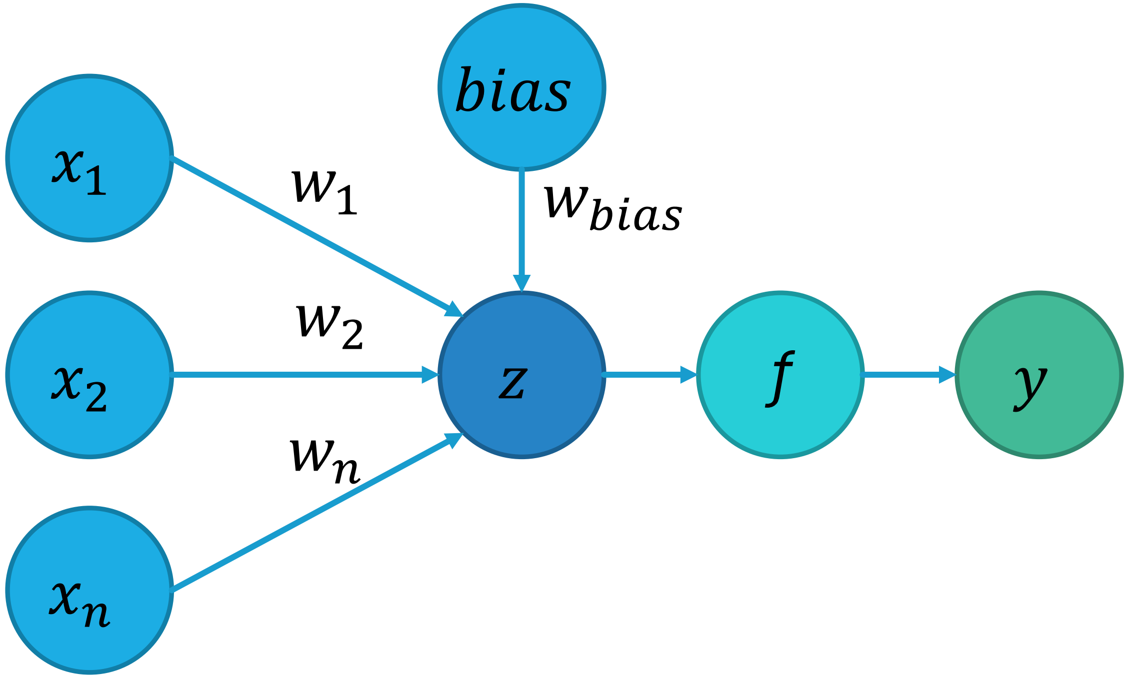
Output

THE PERCEPTRON



THE PERCEPTRON

- Building block of neural networks (single neuron)
- Mimics the neuron in the human brain
- **Forward propagation (input → neuron → output)**



Output, Bias, Inputs

$$y = f\left(bias\ w_{bias} + \sum_{i=1}^{n} x_i\ w_i\right)$$

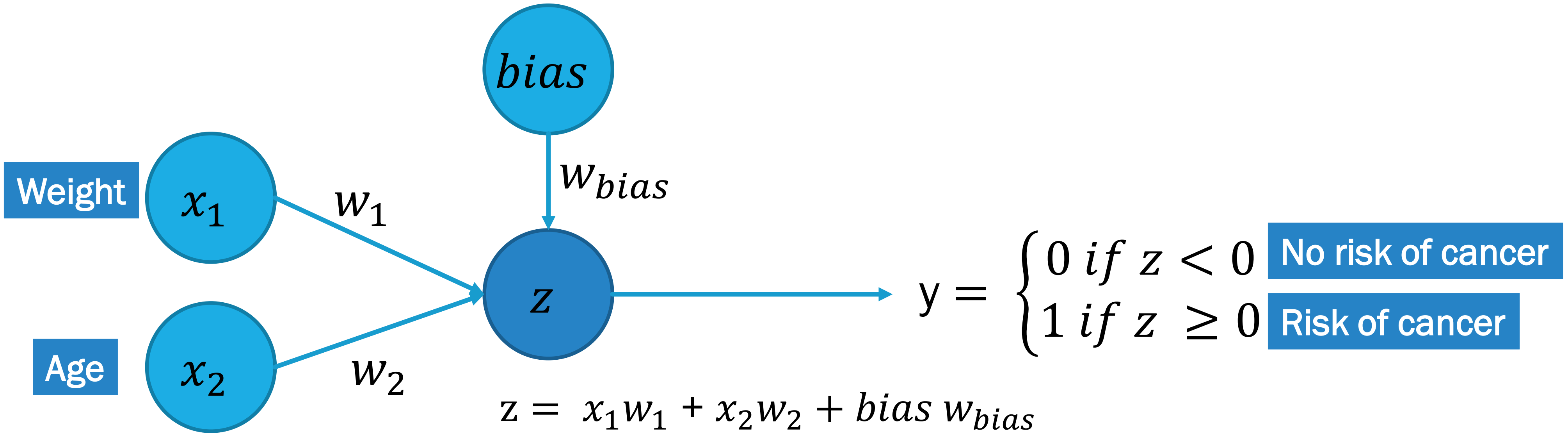
Non-linear function, Sum, Weights

$$z = x_1\ w_1 + x_2\ w_2 + \ldots + x_n\ w_n + bias\ w_{bias}$$

$$y = f(z)$$

$$\text{Example: } y = \begin{cases} 0\ if\ z < 0 \\ 1\ if\ z \geq 0 \end{cases}$$

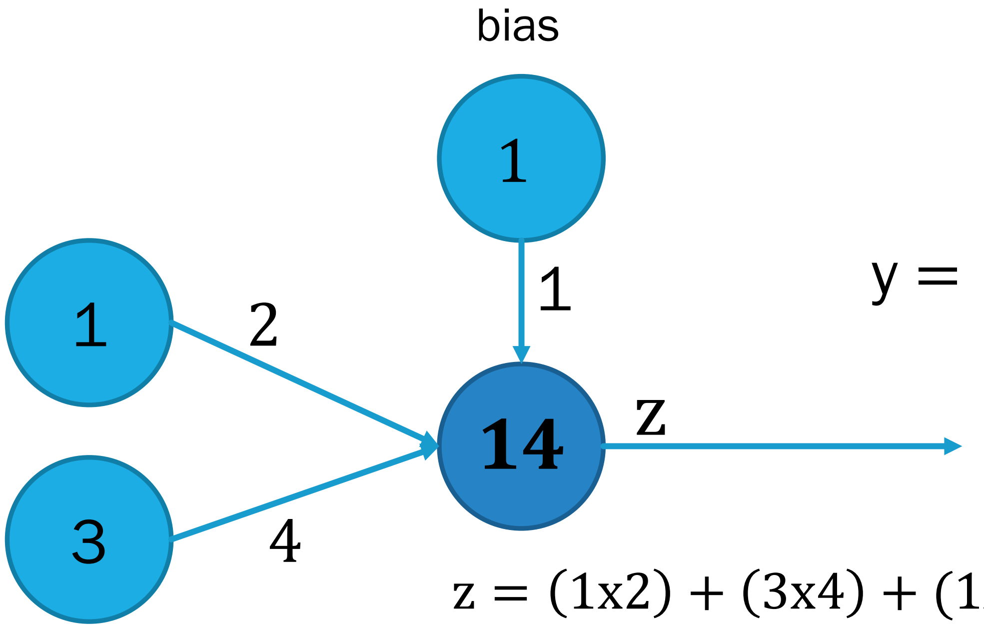
PERCEPTRON: FORWARD PROPAGATION EXAMPLE



Do you know another example of a perceptron?

- Logic operations (**except XOR**)
- Linear seperation
- Predict who is going to pass this module based on the number of lectures and tutorials attended

PERCEPTRON: FORWARD PROPAGATION EXAMPLE

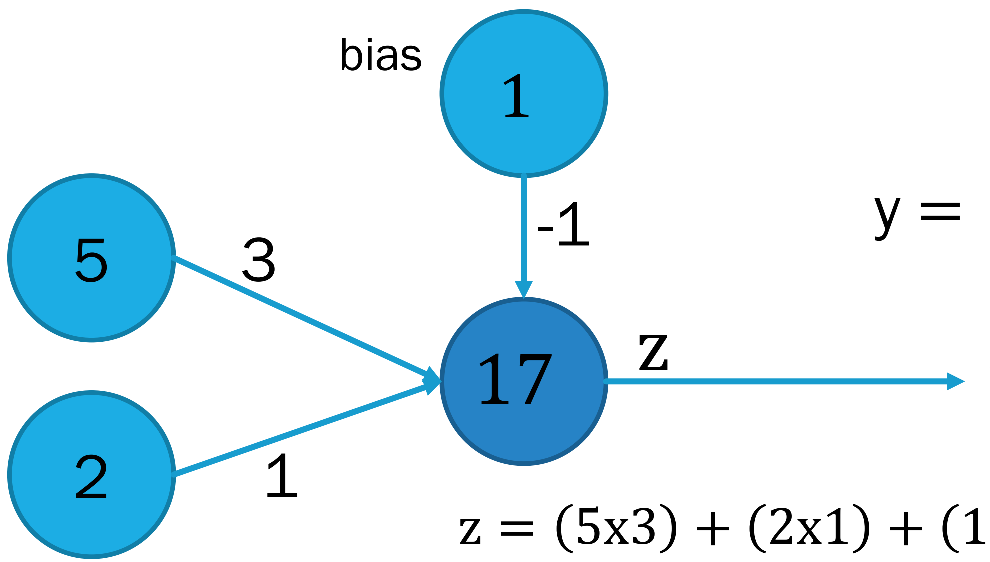


$$y = \begin{cases} 0 \; if \; z < 0 \\ 1 \; if \; z \geq 0 \end{cases}$$

$14 \geq 0; \mathrm{y} = \mathbf{1}$

$$\mathrm{z} = (1\mathrm{x}2) + (3\mathrm{x}4) + (1\mathrm{x}1) =$$
$$= 2 + 12 + 1 = \mathbf{14}$$

PERCEPTRON: FEEDBACK



$$y = \begin{cases} 0 \; if \; z < 20 \\ 1 \; if \; z \geq 20 \end{cases}$$

$$16 < 20; \mathrm{y} = \mathbf{0}$$

$$\mathrm{z} = (5\mathrm{x}3) + (2\mathrm{x}1) + (1\mathrm{x} - 1) =$$
$$= 15 + 2 - 1 = \mathbf{16}$$

Compare answers with the person sitting next to you

Applied AI

pollev.com/esterbonmati

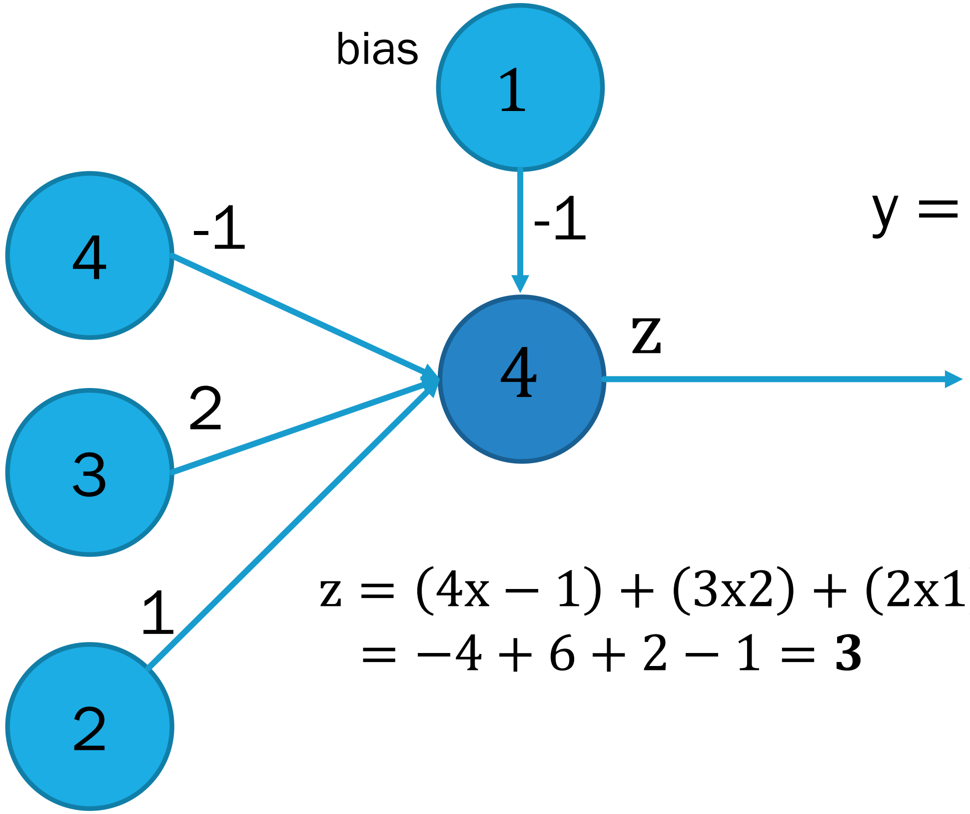
When poll is active respond at PollEv.com/esterbonmati

Result perceptron 1

Nobody has responded yet.

Hang tight! Responses are coming in.

PERCEPTRON: FEEDBACK



$$y = \begin{cases} 0 \ if \ z < 5 \\ 1 \ if \ z \geq 5 \end{cases}$$

$$3 < 5; \mathrm{y} = \mathbf{0}$$

$$z = (4x - 1) + (3x2) + (2x1) + (1x - 1) =$$
$$= -4 + 6 + 2 - 1 = \mathbf{3}$$

Compare answers with the person sitting next to you

Applied AI

pollev.com/esterbonmati

When poll is active respond at PollEv.com/esterbonmati

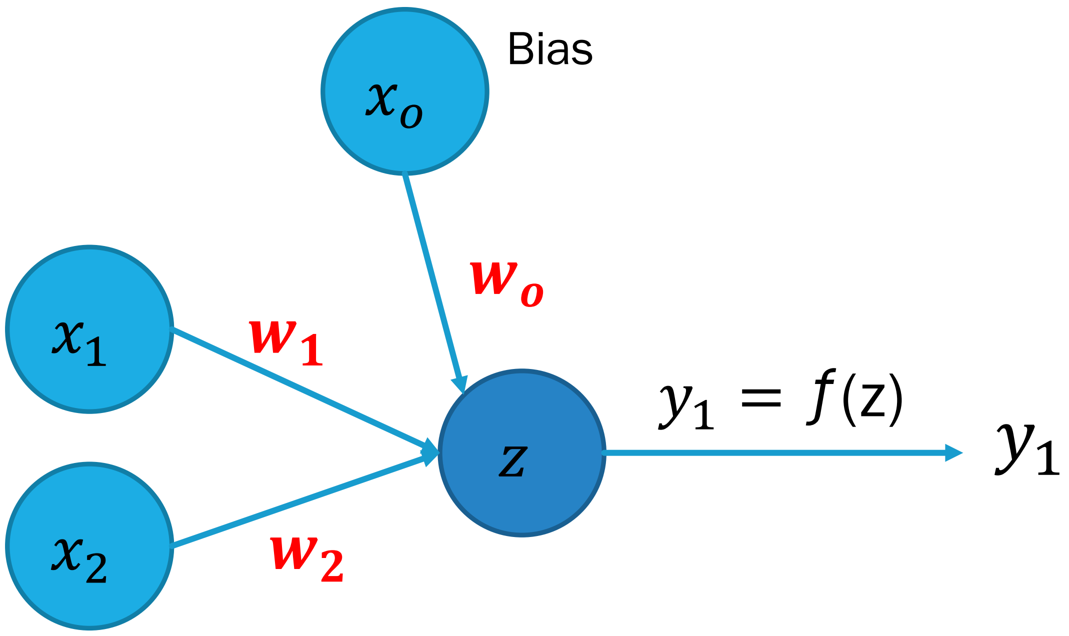
Results perceptron 2

Nobody has responded yet.

Hang tight! Responses are coming in.

PERCEPTRON: HOW DO WE FIND THE WEIGHTS?

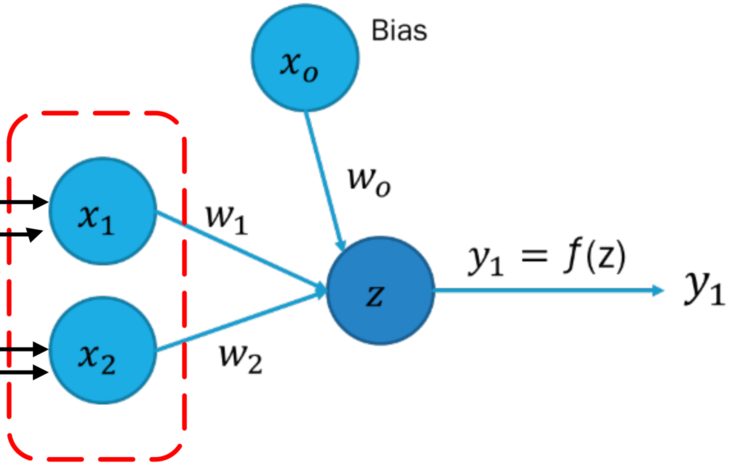
Problem: we want to train a perceptron to classify points in 2 classes.



$$z = x_1 w_1 + x_2 w_2 + x_0 w_0$$

How do we find the weights to classify the points?

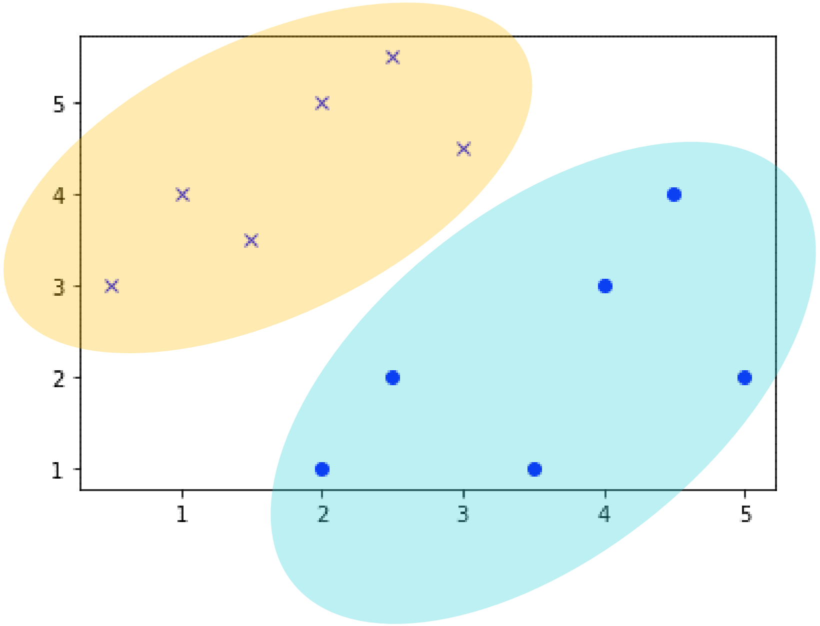
PERCEPTRON: INPUT DEFINITION



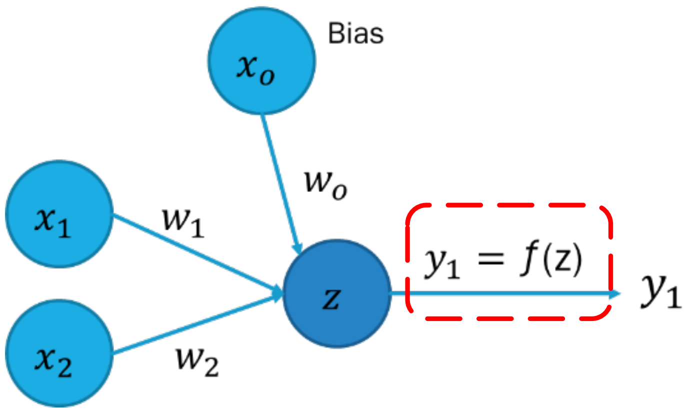
<u>Problem</u>: We have 2 groups of samples, each of them defined by 2 values, and we want to use a perceptron to classify them into their group.

```
x1_group1 = [2, 5, 4, 2.5, 3.5, 4.5]
x2_group1 = [1, 2, 3, 2, 1, 4]

x1_group0 = [0.5, 2.5, 3, 1.5, 1, 2]
x2_group0 = [3, 5.5, 4.5, 3.5, 4, 5]
```



PERCEPTRON: ACTIVATION FUNCTION



$$y = \begin{cases} -1 \; if \; z \leq 0 \\ 1 \; if \; z > 0 \end{cases}$$

If the sum is is greater than 0 we return 1, otherwise we return 0

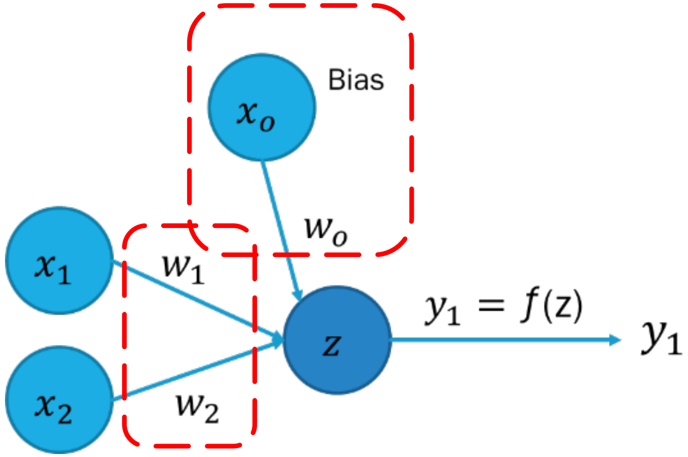
In [112]:
```
def activation_function(value):
    if value > 0: return 1
    else: return -1
```

PERCEPTRON: WEIGHTS AND BIAS DEFINITION

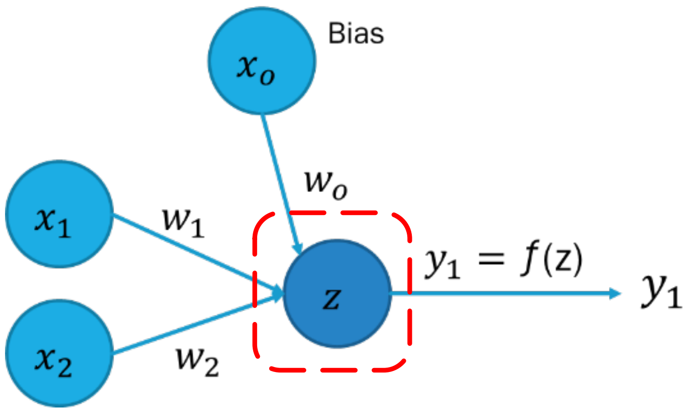
Some random weights…

```
w1 = 0.75
w2 = -0.5
bias = 1
wbias = 1
```

How good are these weights in grouping the input data?



PERCEPTRON: FORWARD PROPAGATION



We need to find z in order to calculate the output y …

```
def perceptron(x1, x2, bias):
    z = x1*w1 + x2*w2 + bias*wbias
    return z
```

Then we can calculate the output y using the perceptron and the activation function:

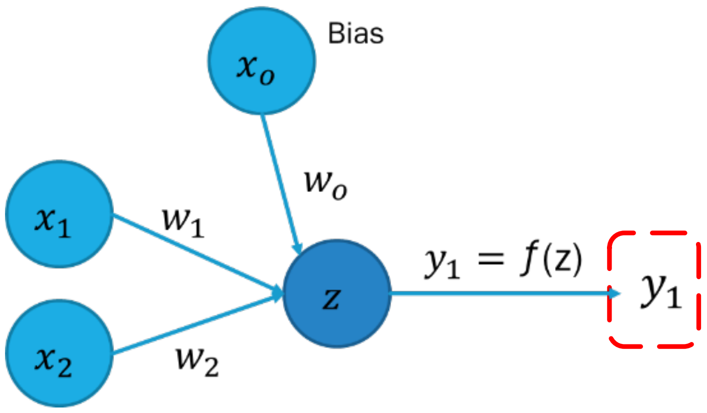
```
for i in range(len(x1_group0)):
    z = perceptron(x1_group0[i], x2_group0[i], bias)
    group = activation_function(z)
    print('Point ' + str(i) + ' in group 0 returned ' + str(group))
```

PERCEPTRON: PREDICTION AND RESULTS

We calculate the output for each group, so we know which is the prediction:

```
    group = activation_function(z)
    print('Point ' + str(i) + ' in group 1 returned ' + str(group))

for i in range(len(x1_group0)):
    z = perceptron(x1_group0[i], x2_group0[i], bias)
    group = activation_function(z)
    print('Point ' + str(i) + ' in group 0 returned ' + str(group))
```




```
Point 0 in group 1 returned 1
Point 1 in group 1 returned 1
Point 2 in group 1 returned 1
Point 3 in group 1 returned 1
Point 4 in group 1 returned 1
Point 5 in group 1 returned 1
Point 0 in group 0 returned -1
Point 1 in group 0 returned 1
Point 2 in group 0 returned 1
Point 3 in group 0 returned 1
Point 4 in group 0 returned -1
Point 5 in group 0 returned -1
```

Incorrect

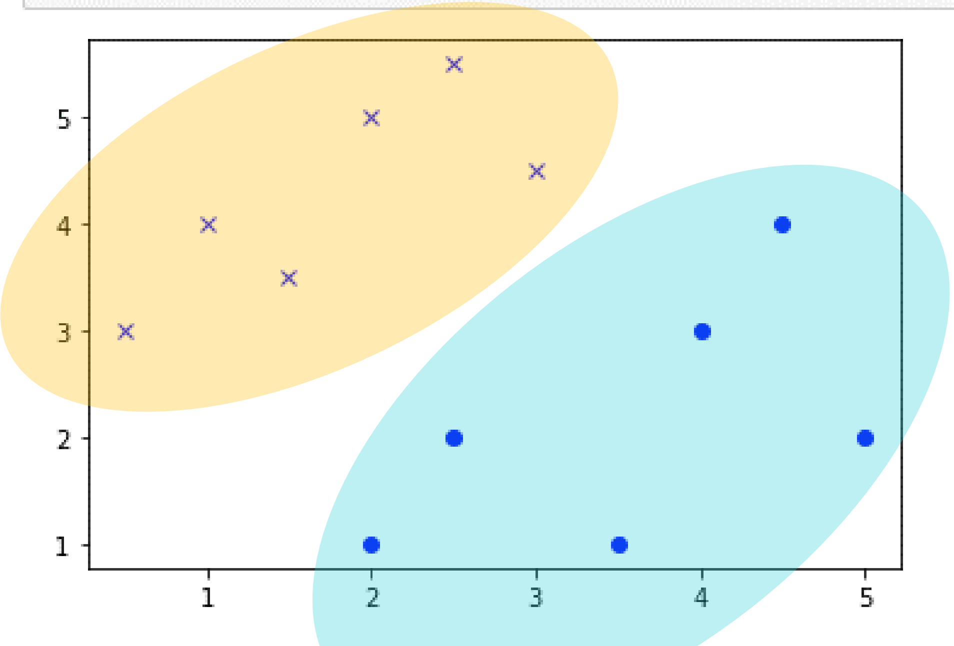
$$accuracy = \frac{number\ correct\ samples}{total\ number\ of\ samples} = \frac{9}{12} = 0.75 \rightarrow \mathbf{75}\ \%$$

Can we improve the accuracy if we change the weights?

PERCEPTRON: VISUALISATION

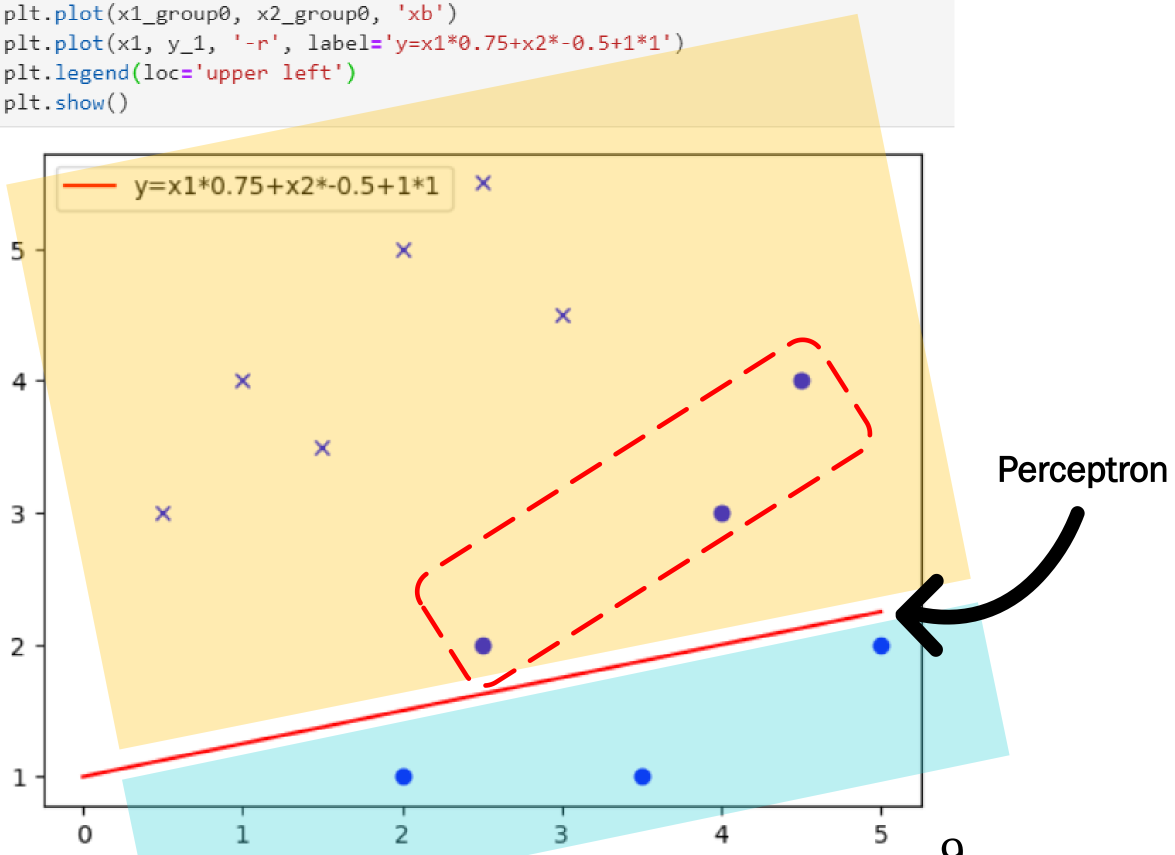
Plot the points

In [117]:
```
fig = plt.figure()
ax = plt.axes()
plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.show()
```



[7]:
```
x1 = np.linspace(0,5,100)
x2 = np.linspace(0,5,100)
y_1 = w1*x1 + w2*x2 + 1

plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.plot(x1, y_1, '-r', label='y=x1*0.75+x2*-0.5+1*1')
plt.legend(loc='upper left')
plt.show()
```



$$y = x_1 w_1 + x_2 w_2 + bias\ w_{bias}$$

$$accuracy = \frac{9}{12} = 0.75$$

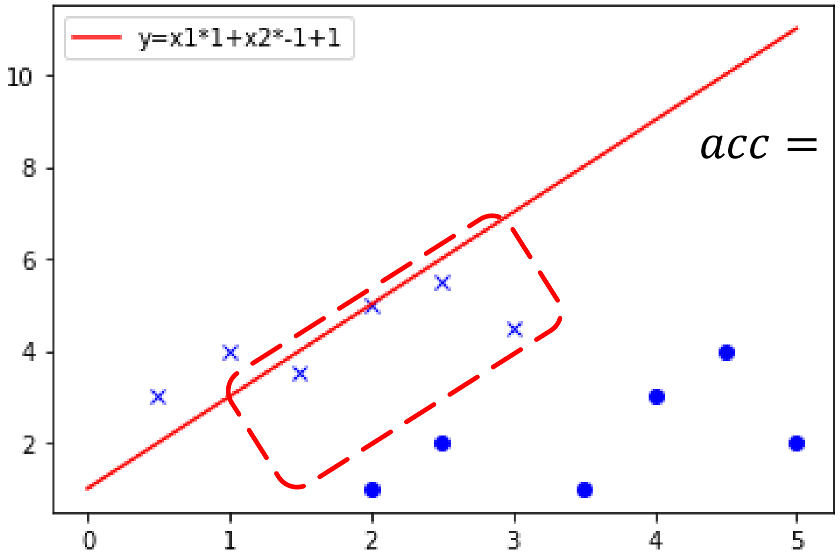
PERCEPTRON: UPDATE WEIGHTS

To change the prediction output, we need to change the weights

In [102]:
```
w1 = 1
w2 = 1
bias = 1
wbias = 1

y_1 = w1*x1 + w2*x2 + bias*wbias

plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.plot(x1, y_1, '-r', label='y=x1*1+x2*1+1*1')
plt.legend(loc='upper left')
plt.show()
```

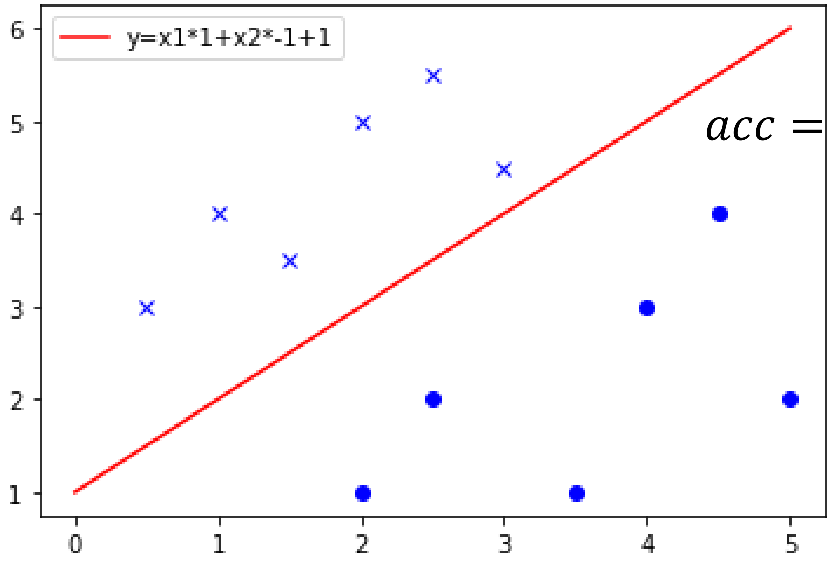


$$acc = \frac{8}{12} = 0.67$$

In [103]:
```
w1 = 3
w2 = -2
bias = 1
wbias = -1

y_1 = x1*w1 + x2*w2 + bias*bias

plt.plot(x1_group1, x2_group1, 'ob')
plt.plot(x1_group0, x2_group0, 'xb')
plt.plot(x1, y_1, '-r', label='y=x1*3+x2*-2+1*-1')
plt.legend(loc='upper left')
plt.show()
```



$$acc = \frac{12}{12} = 1.0$$

PERCEPTRON: NEW PREDICTION

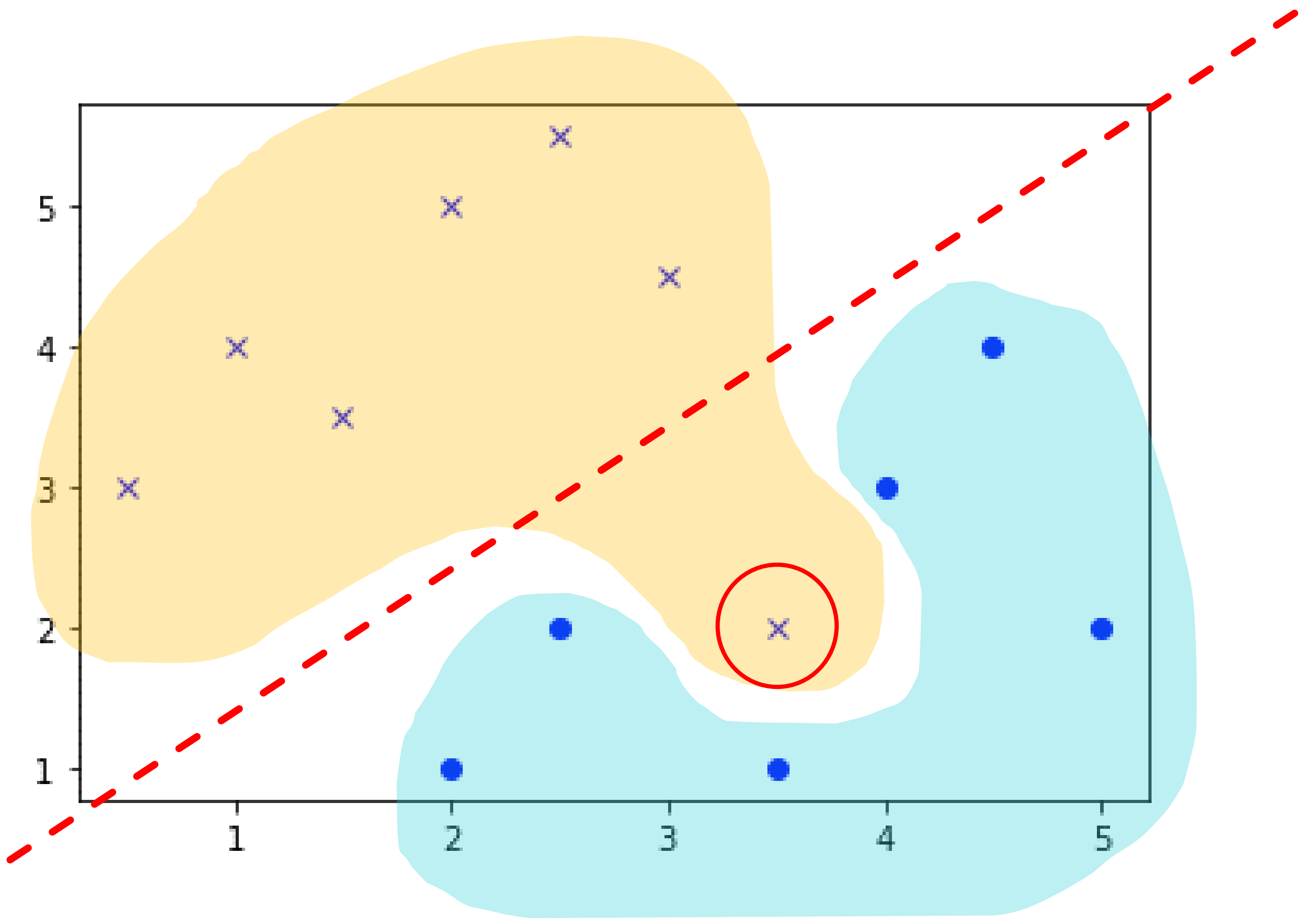
In [104]:
```
for i in range(len(x1_group1)):
    sum = perceptron(x1_group1[i], x2_group1[i], bias)
    group = activation_function(sum)
    print('Point ' + str(i) + ' in group 1 returned ' + str(group))

for i in range(len(x1_group0)):
    sum = perceptron(x1_group0[i], x2_group0[i], bias)
    group = activation_function(sum)
    print('Point ' + str(i) + ' in group 0 returned ' + str(group))
```

```
Point 0 in group 1 returned 1
Point 1 in group 1 returned 1
Point 2 in group 1 returned 1
Point 3 in group 1 returned 1
Point 4 in group 1 returned 1
Point 5 in group 1 returned 1
Point 0 in group 0 returned -1
Point 1 in group 0 returned -1
Point 2 in group 0 returned -1
Point 3 in group 0 returned -1
Point 4 in group 0 returned -1
Point 5 in group 0 returned -1
```

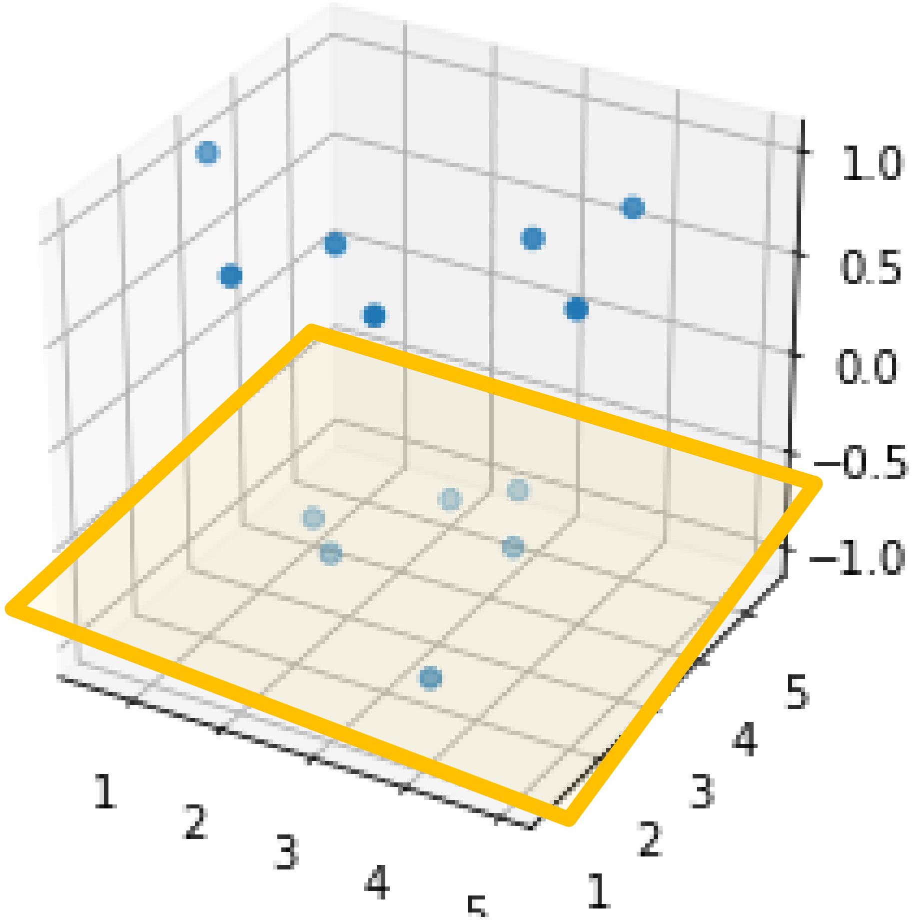
$$acc = \frac{12}{12} = 1.0 \rightarrow 100\ \%$$

NEURAL NETWORKS: ANOTHER EXAMPLE

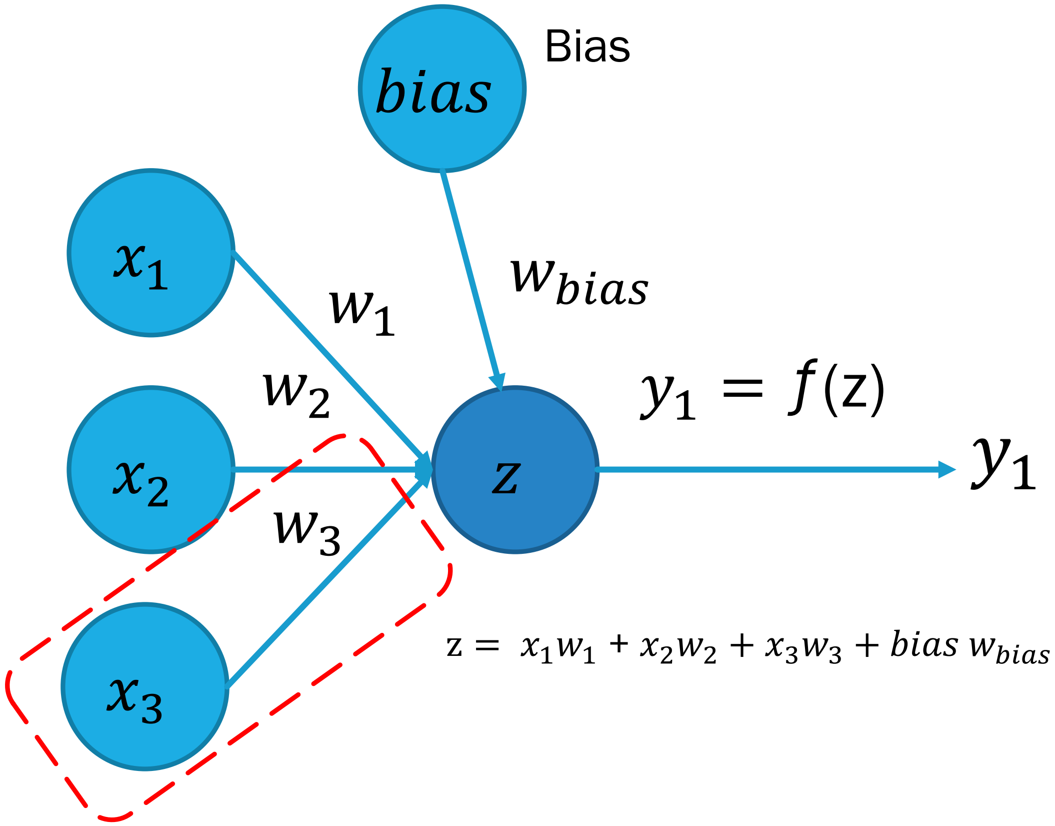


NEURAL NETWORKS: ANOTHER EXAMPLE

$$y = x_1w_1 + x_2w_2 + \boxed{x_3w_3} + b$$



Bias



$\mathrm{z} = x_1w_1 + x_2w_2 + x_3w_3 + bias\ w_{bias}$

[For reference only] Equation of a plane = $ax + by + cz + d$

pollev.com/esterbonmati

Join by Web PollEv.com/esterbonmati

How many inputs can a perceptron have?

2

0%

3

SEE MORE

pollev.com/esterbonmati

When poll is active respond at PollEv.com/esterbonmati

How many weights has each input in a perceptron?

0

0%

1

SEE MORE

Applied AI

Moderate Visual settings Edit

pollev.com/esterbonmati

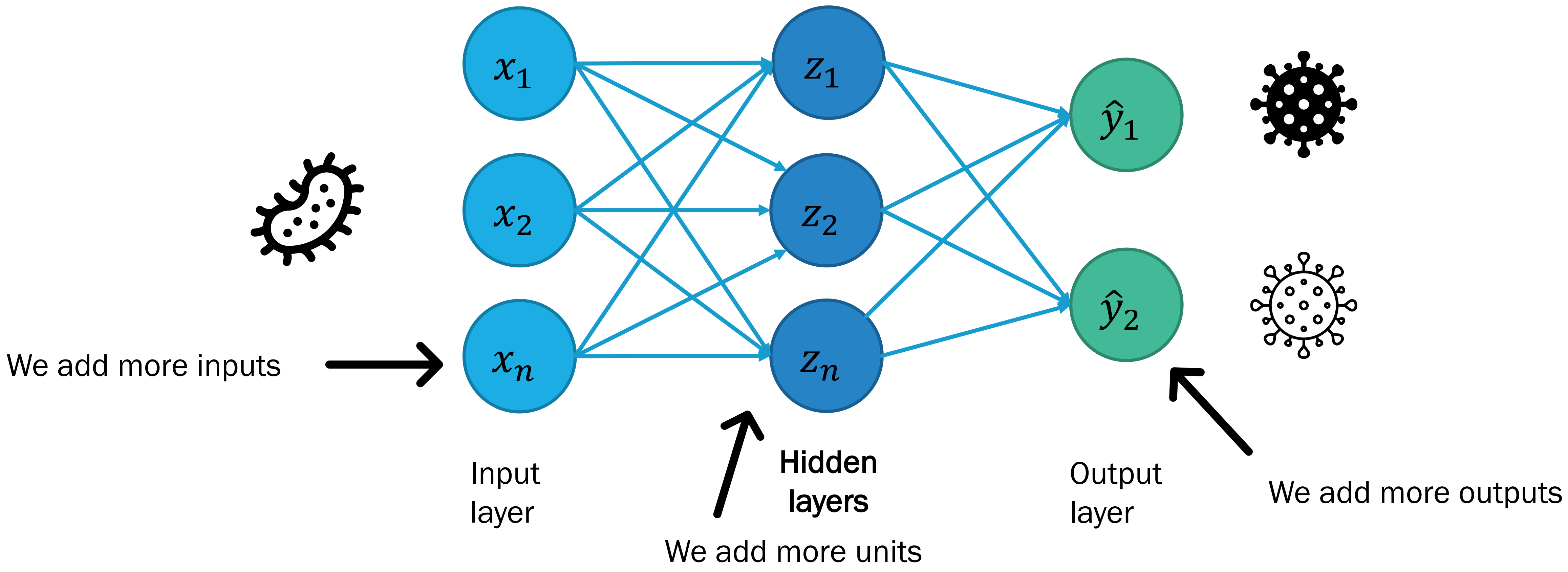
When poll is active respond at PollEv.com/esterbonmati

Q&A

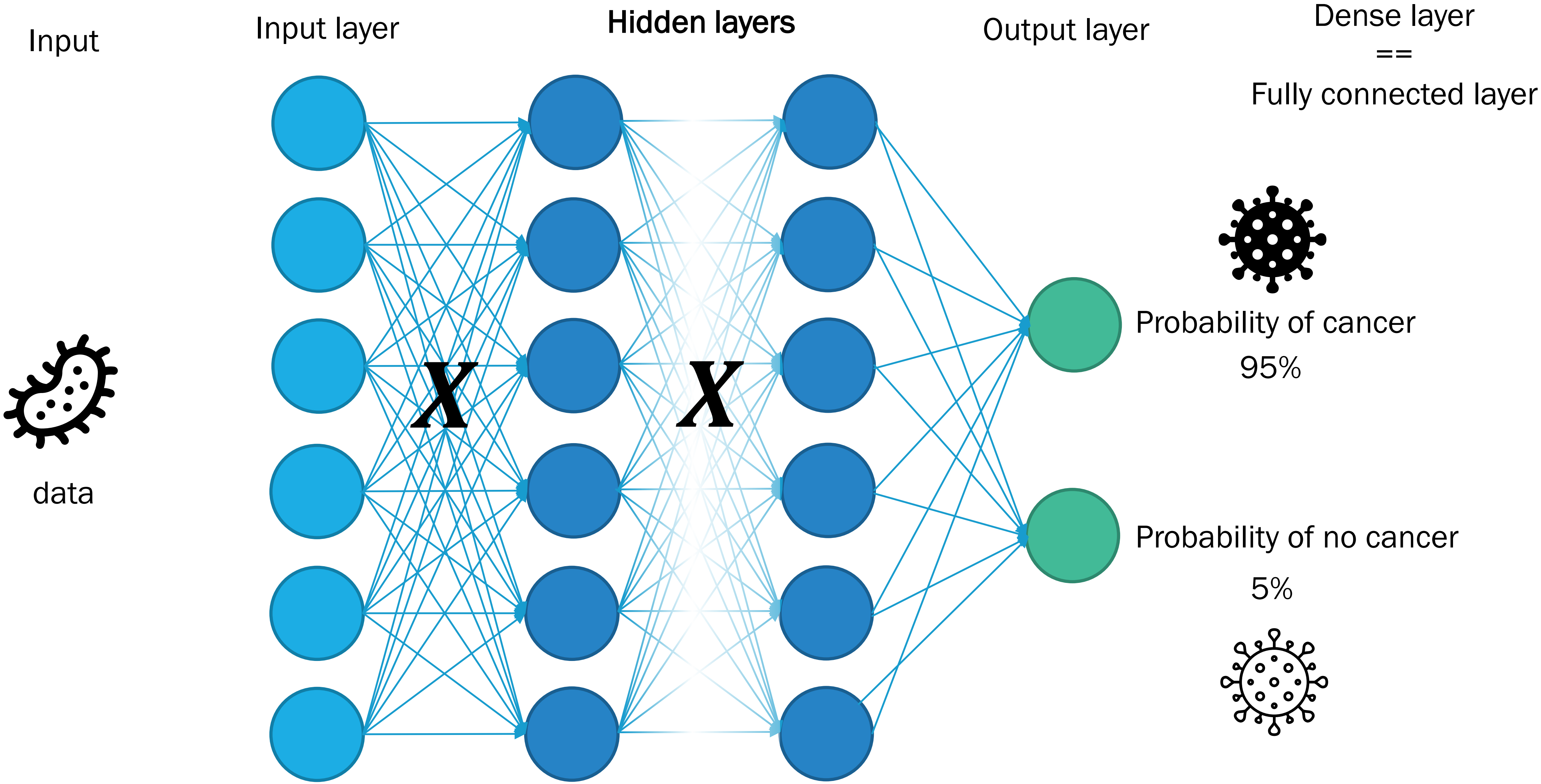
Nobody has responded yet.

Hang tight! Responses are coming in.

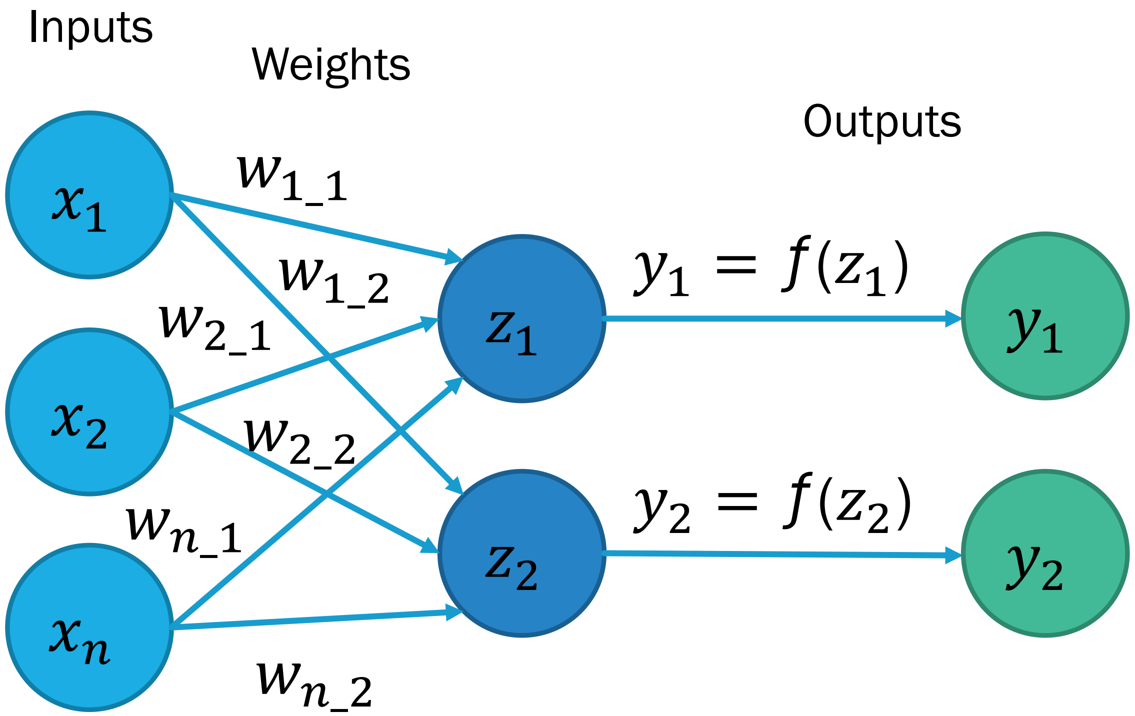
MULTI-LAYER NEURAL NETWORKS



MULTI-LAYER NEURAL NETWORKS EXAMPLE



ARTIFICIAL NEURAL NETWORKS (ANN)

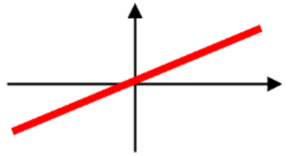
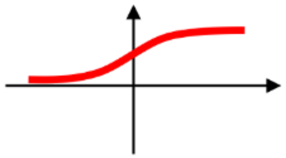
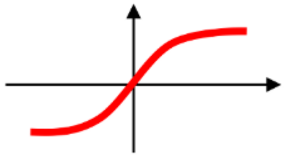
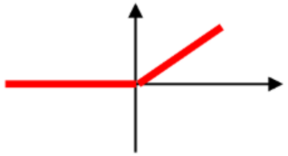


Each output has its own weight for each input

$$\mathrm{z} = w_0 + \sum_{i=1}^{n} x_i\, w_i$$

$$\hat{y} = f\left(w_0 + \sum_{i=1}^{n} x_i\, w_i\right)$$

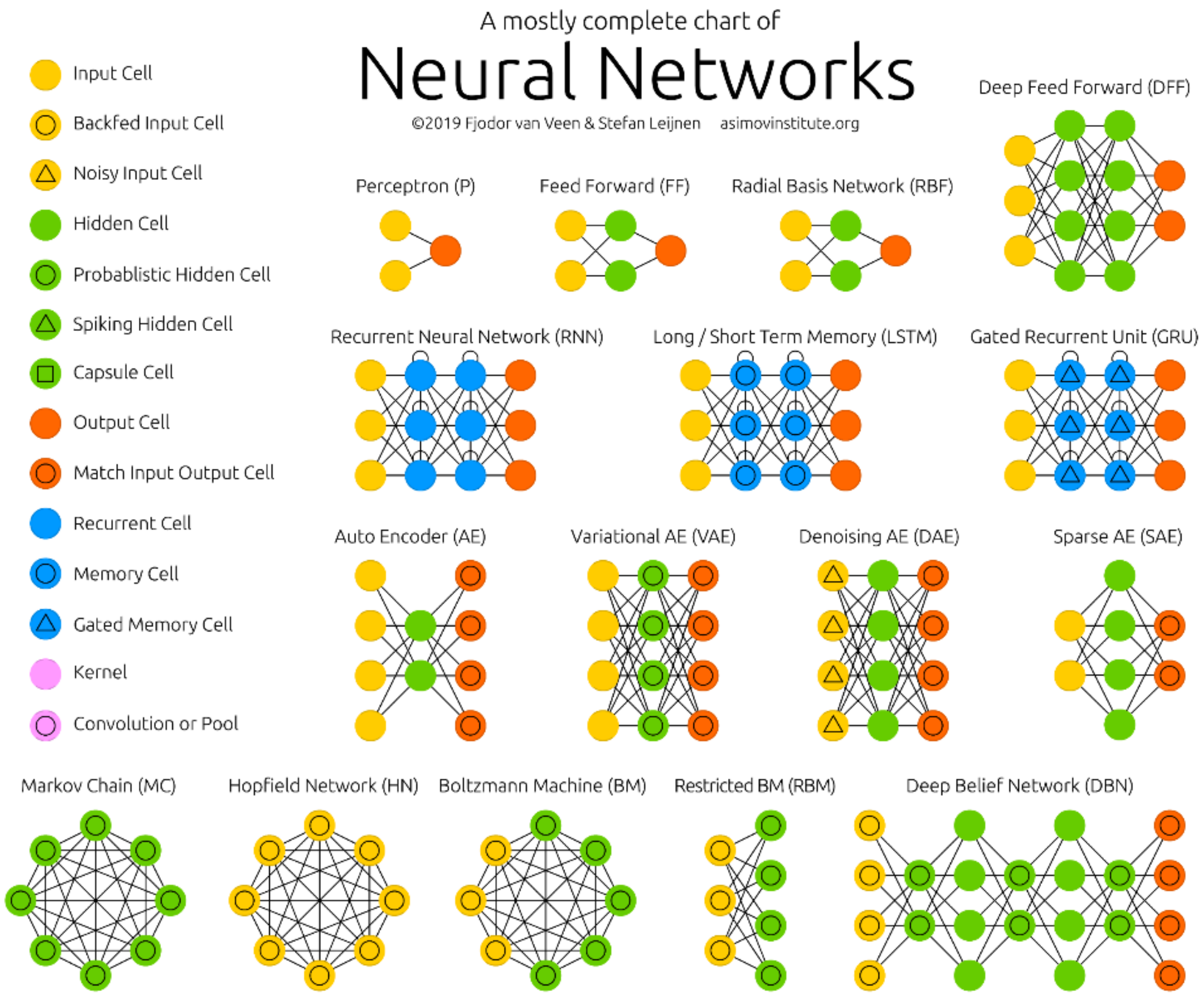
FOR REFERENCE ONLY: ACTIVATION FUNCTIONS

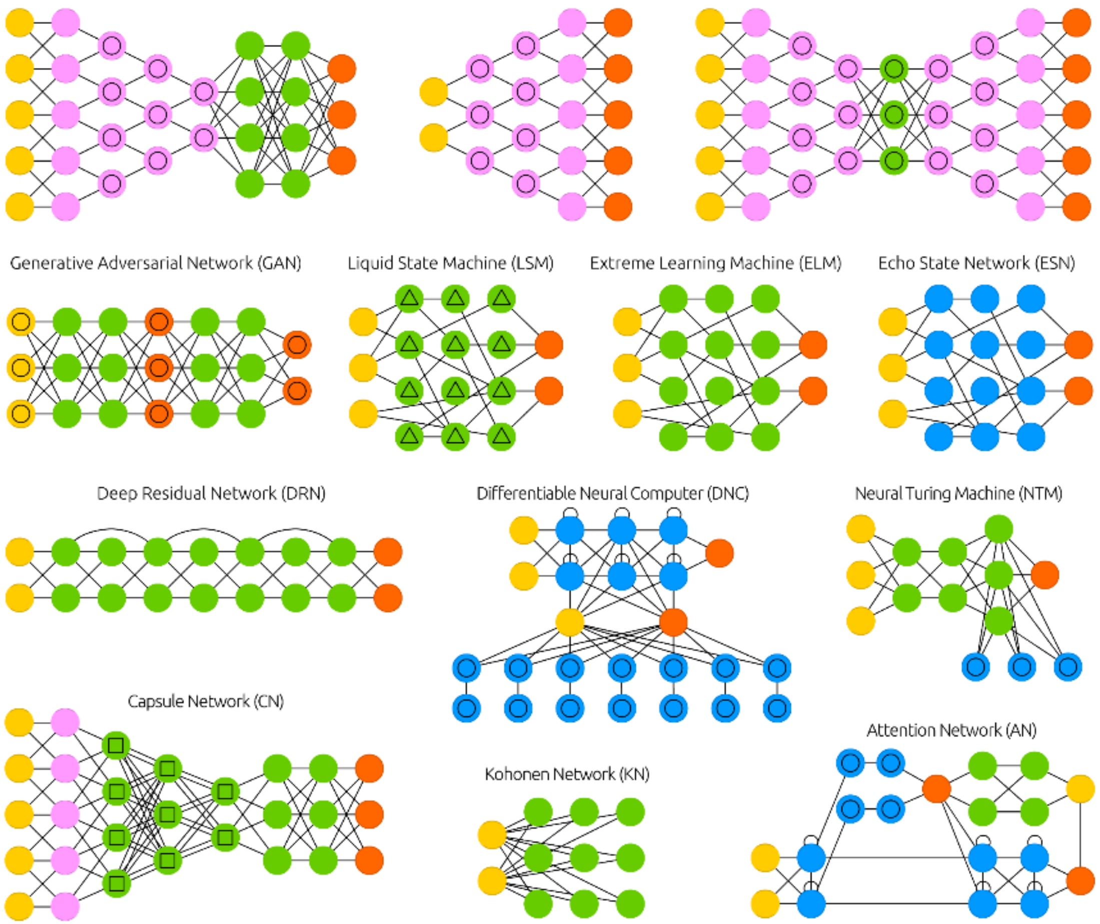
Activation function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent (Tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	

Non-linear

Non-linear activation functions allow to learn more complex mappings

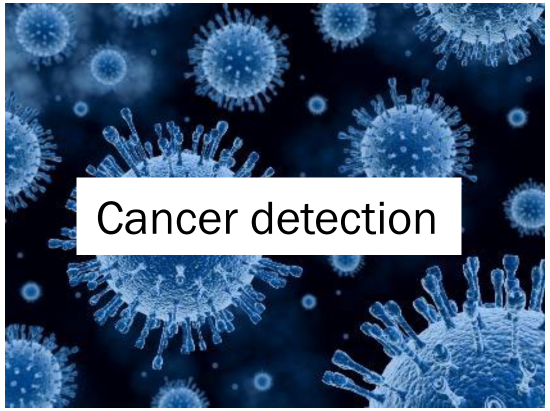
EXAMPLES OF NEURAL NETWORKS





Pictures from: https://www.asimovinstitute.org/neural-network-zoo/

WHAT CAN WE DO WITH ANN?


Cancer detection


Shopping prediction


Recognition/Classification

Interests prediction


Chatbots


Drug discovery


Weather prediction


Stock market prediction

Do you know more examples?

RESEARCH

Forecasting Climatic Trends Using Neural Networks: An Experimental Study Using Global Historical Data

Takeshi Ise[1,2] and Yurika Oba[1]*

[1] Field Science Education and Research Center (FSERC), Kyoto University, Kyoto, Japan, [2] Japan Science and Technology Agency (JST), Kawaguchi, Japan

Review Article

Artificial neural networks in the cancer genomics frontier

Andrew Oustimov[1], Vincent Vu[2]

[1]Department of Epidemiology & Biostatistics, College of Public Health, University of South Florida, Tampa, FL 33620, USA; [2]Department of Mathematics and Statistics, University of California, Los Angeles, CA, USA

Correspondence to: Andrew Oustimov, MPH. Department of Epidemiology & Biostatistics, College of Public Health, University of South Florida, 13201 Bruce B Downs Blvd, Tampa, FL 33620, USA. Email: aoustimo@mail.usf.edu.

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 32, NO. 10, OCTOBER 2021 4291

Attention in Natural Language Processing

Andrea Galassi, Marco Lippi, and Paolo Torroni

AUTOMATIC LANGUAGE IDENTIFICATION USING DEEP NEURAL NETWORKS

Ignacio Lopez-Moreno[1], Javier Gonzalez-Dominguez[1,2], Oldrich Plchot[3], David Martinez[4], Joaquin Gonzalez-Rodriguez[2], Pedro Moreno[1]

[1]Google Inc., New York, USA
[2]ATVS-Biometric Recognition Group, Universidad Autonoma de Madrid, Spain
[3]Brno University of Technology, Czech Republic
[4]Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain
{elnota, jgd}@google.com

Tran *et al. Genome Medicine* (2021) 13:152
https://doi.org/10.1186/s13073-021-00968-x

Genome Medicine

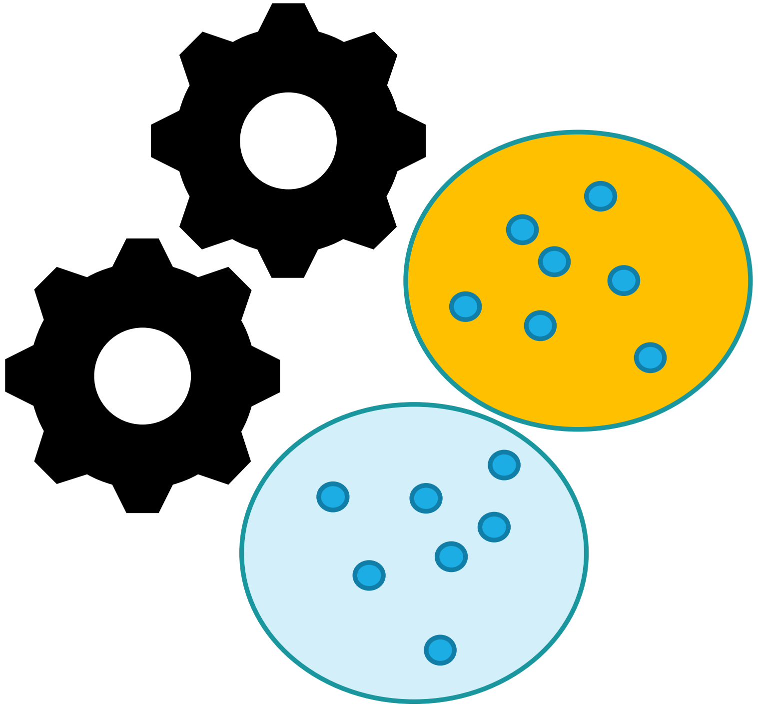
REVIEW **Open Access**

Check for updates

Deep learning in cancer diagnosis, prognosis and treatment selection

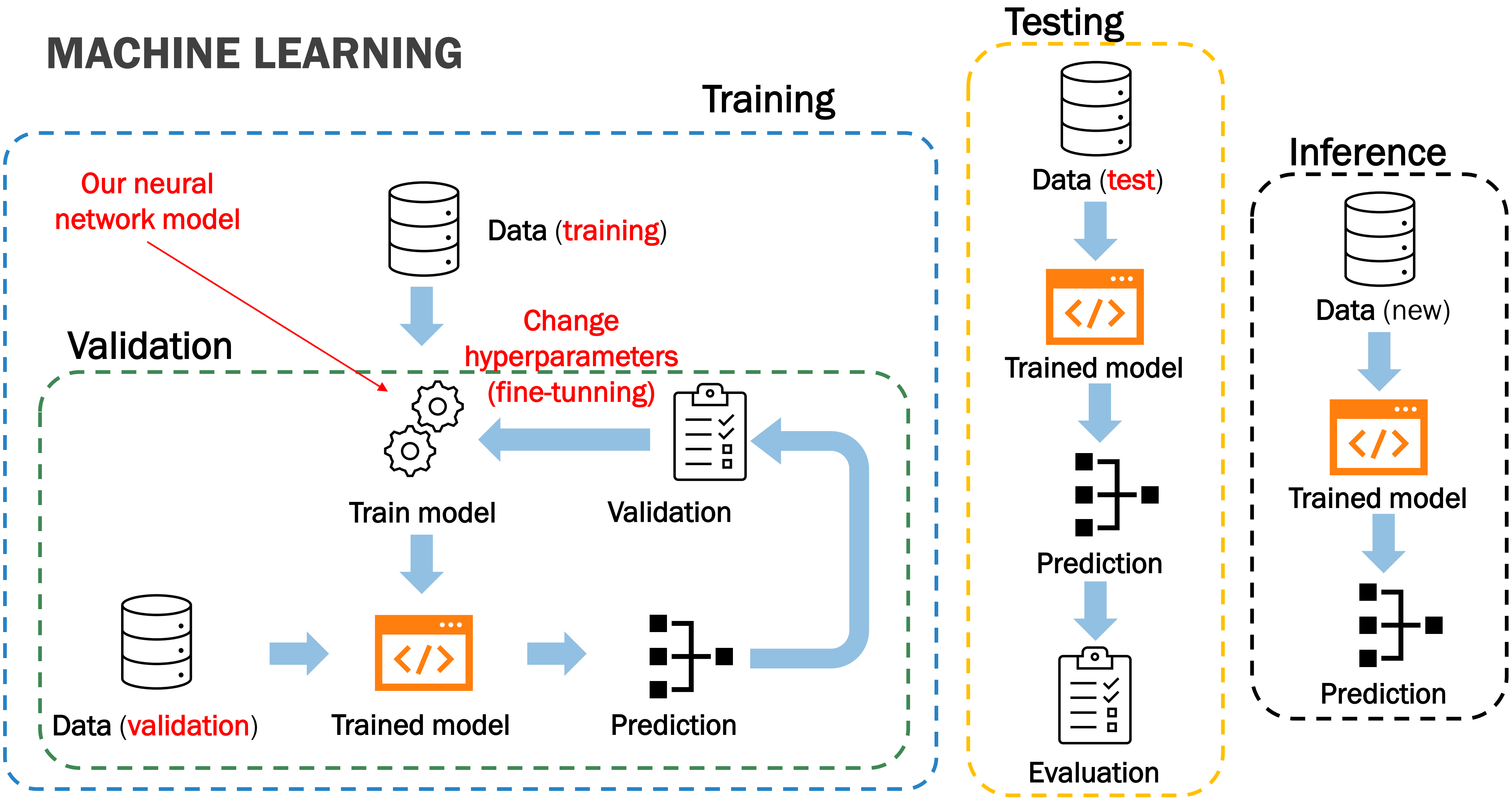
Khoa A. Tran[1,2], Olga Kondrashova[1], Andrew Bradley[4], Elizabeth D. Williams[2,3], John V. Pearson[1] and Nicola Waddell[1*]

SUPERVISED VS UNSUPERVISED



We have both **supervised** and **unsupervised** neural networks. We will focus the rest of the lecture on **supervised** learning.

MACHINE LEARNING

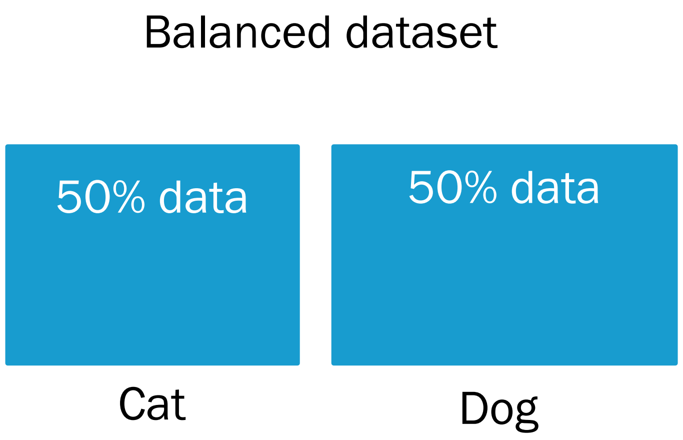


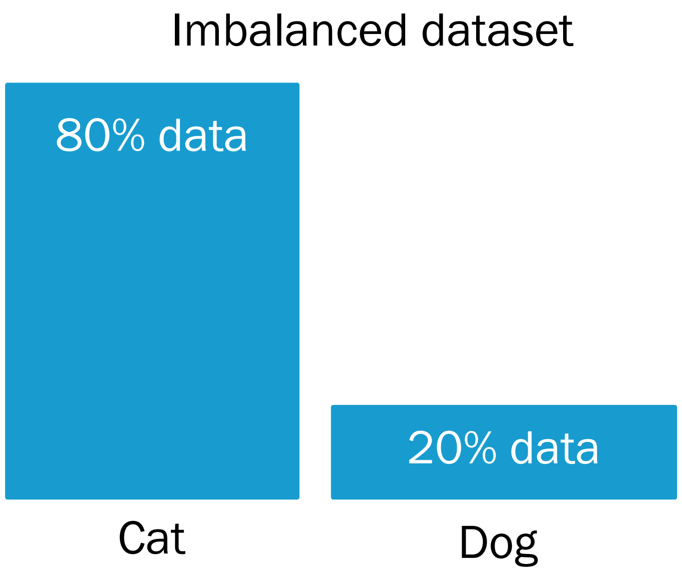
DATA SPLIT

- Data split

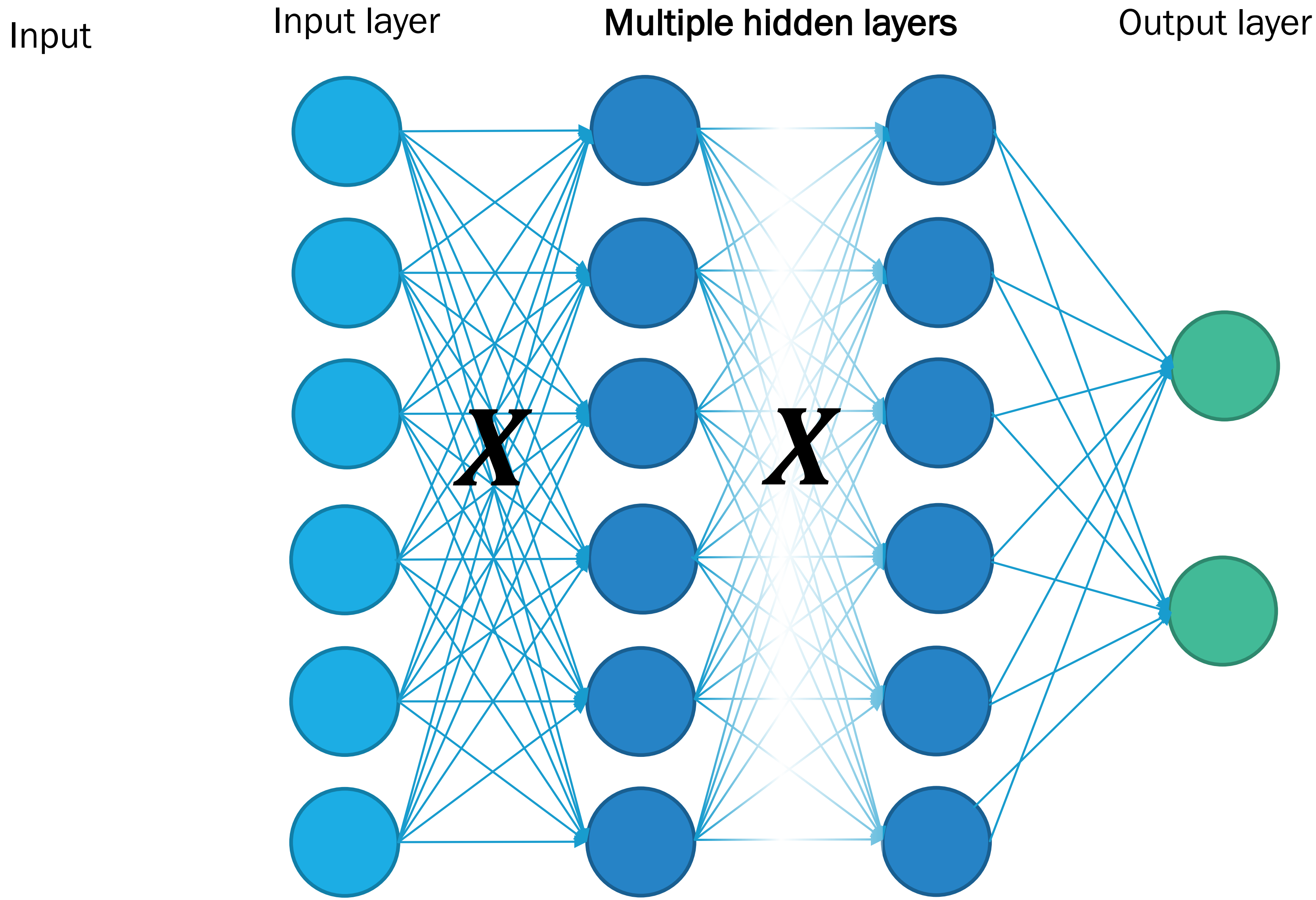
Training set	Validation set	Test set	
80%	10%	10%	(small datasets)
60%	20%	20%	(larger datasets)

- Know your data:





NEURAL NETWORK EXAMPLE



How do we initialise weights?

How can we find the optimum weights so that the error is minimal?

TRAINING A NEURAL NETWORK

Train model

- Objective: find the optimum weights that will give the lowest **error** [loss]
- Optimisation (Gradient Descent)

  1. Randomly initialise weights
  2. Find how much the weights need to change (by computing the derivates)
  3. Update weights by taking a small step [learning rate]
  4. Repeat until convergence (until there is no improvement) [number of epochs]

FOR REFERENCE ONLY: LOSS OPTIMISATION

Train model

- **Identify (train) a set of weights that will give us the minimum error**
- Loss functions: Cross entropy, Mean Squared Error, Mean Absolute Error.
- Needs to be differentiable (the derivative of the function exists for all points).

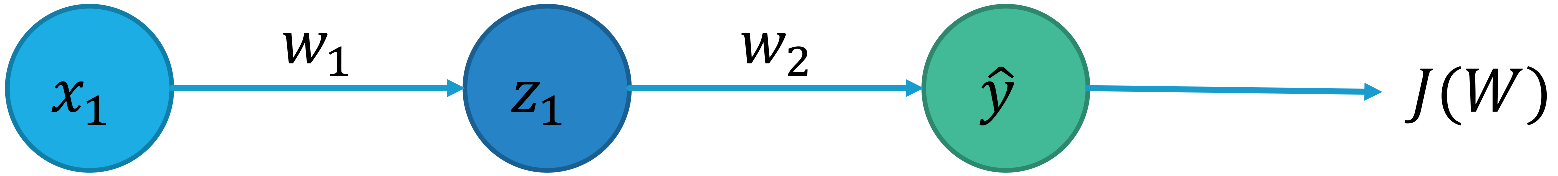
Binary classification	Multiclass classification	Regression
Loss: Cross Entropy	Loss: Cross Entropy	Loss: Mean Squared Error
Activation: Sigmoid	Activation: Softmax	

BACKPROPAGATION: BACKWARD PROPAGATION OF ERRORS

Train model

- Compute the gradients (computationally expensive!)

How does a small change on the weights affect the final loss $J(W)$?



Watch: https://www.youtube.com/watch?v=IN2XmBhILt4&list=PLblh5JKOoLUIxGDQs4LFFD--41Vzf-ME1&index=4

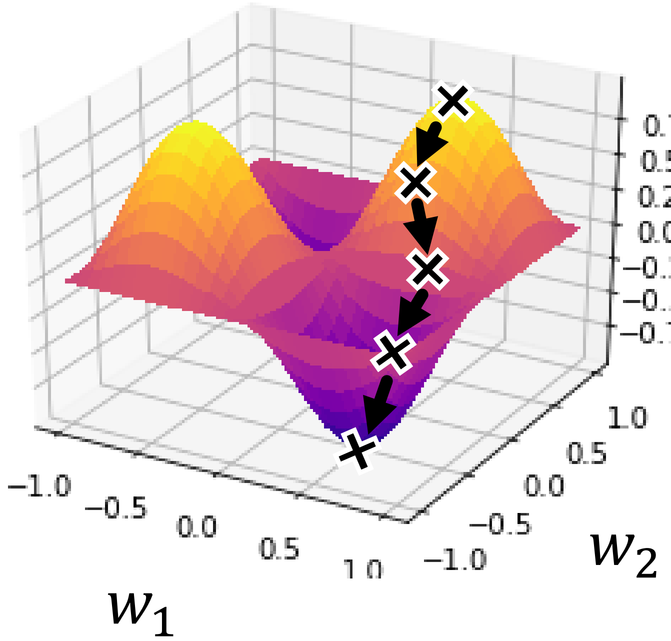
How much a change in w_2 will affect the total error (loss) $J(W)$?

The partial derivative of the loss with respect to w_2

$$\frac{\partial J(W)}{\partial w_2} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2}$$

How much a change in w_1 will affect the total error (loss) $J(W)$?

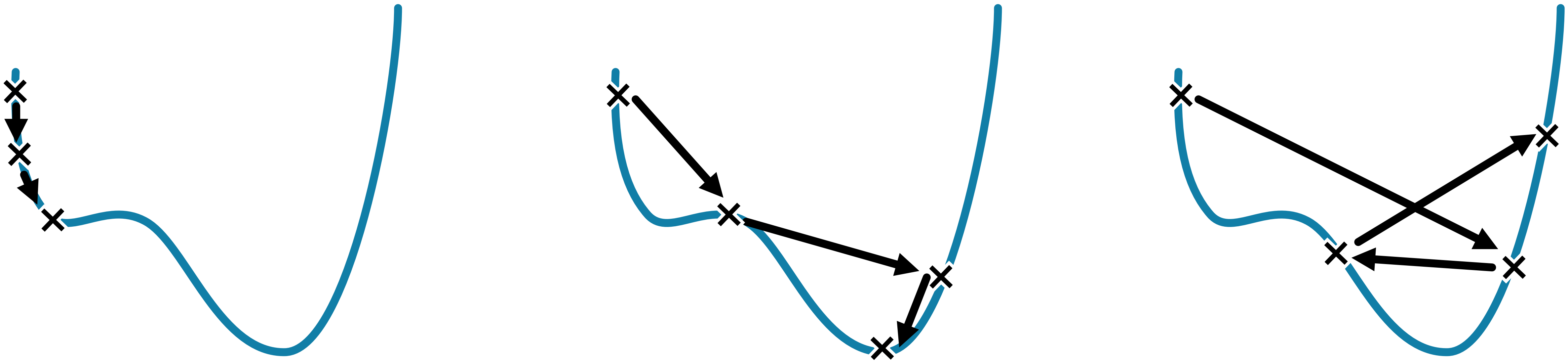
$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$



LEARNING RATE

Train model

- How much step we take in the gradient?



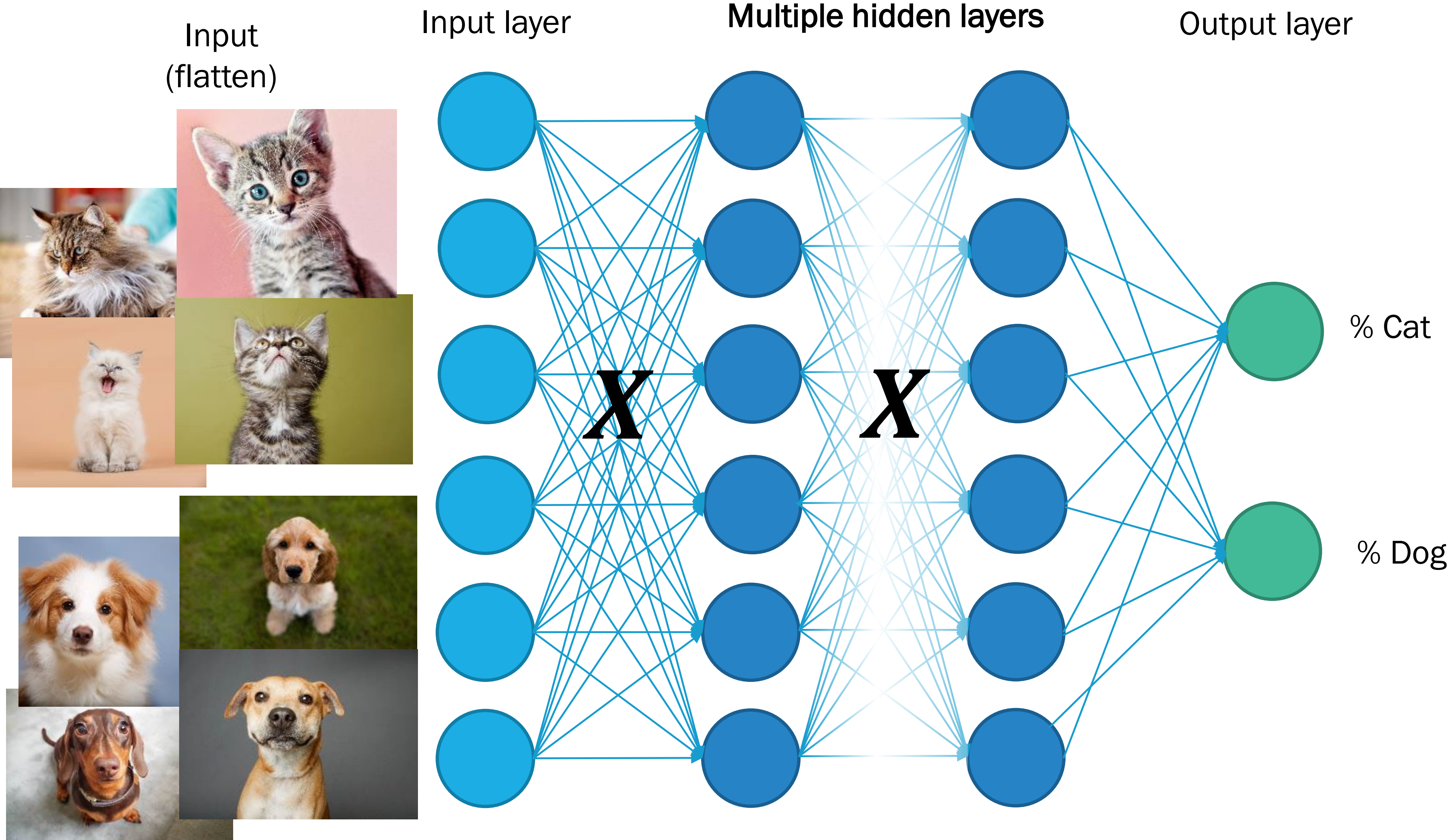
Small learning rate

Good learning rate

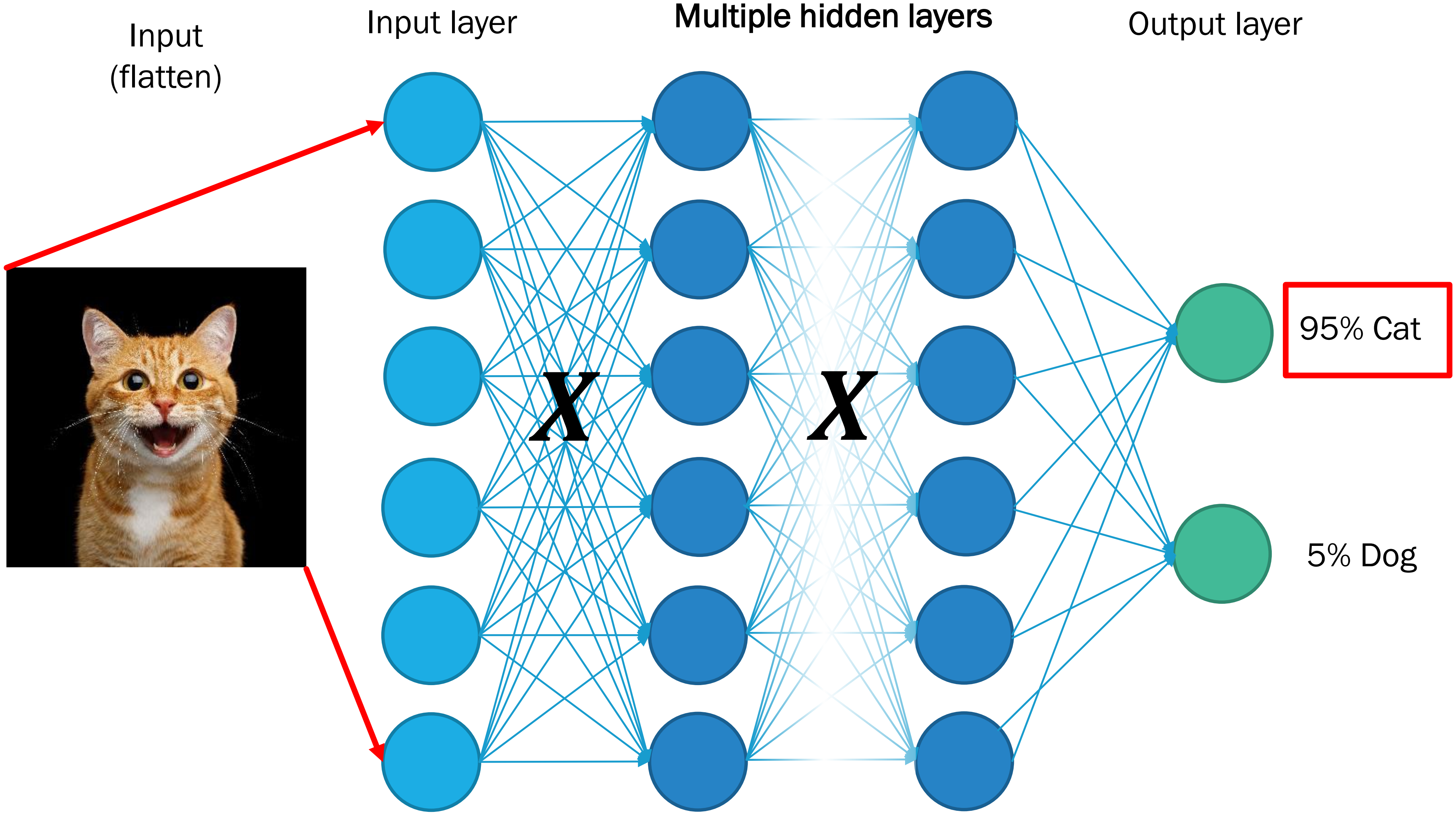
Large learning rate

FORWARD PROPAGATION AND BACKPROPAGATION

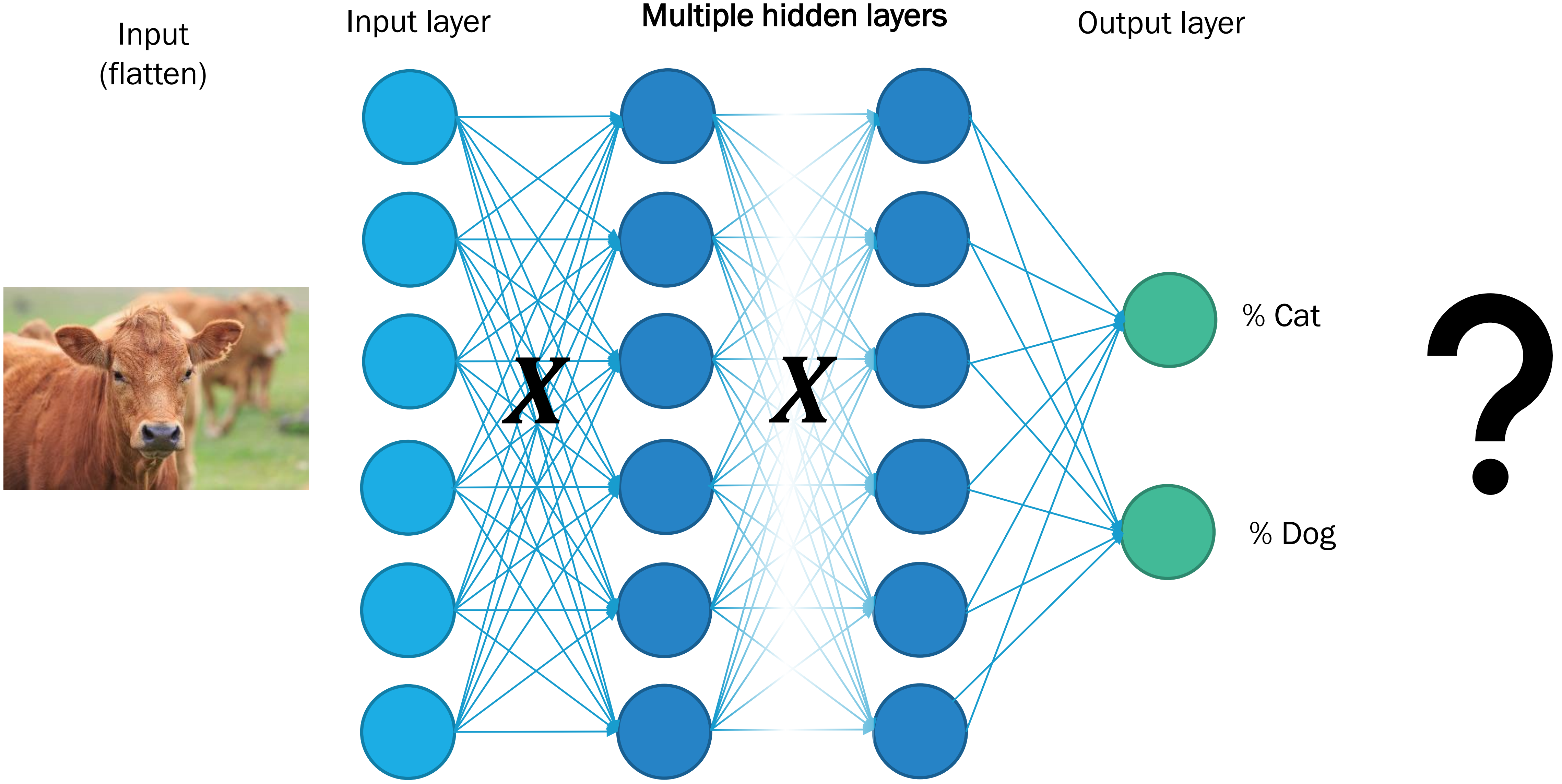
Train model



FORWARD PROPAGATION AND BACKPROPAGATION



FORWARD PROPAGATION AND BACKPROPAGATION



Applied AI

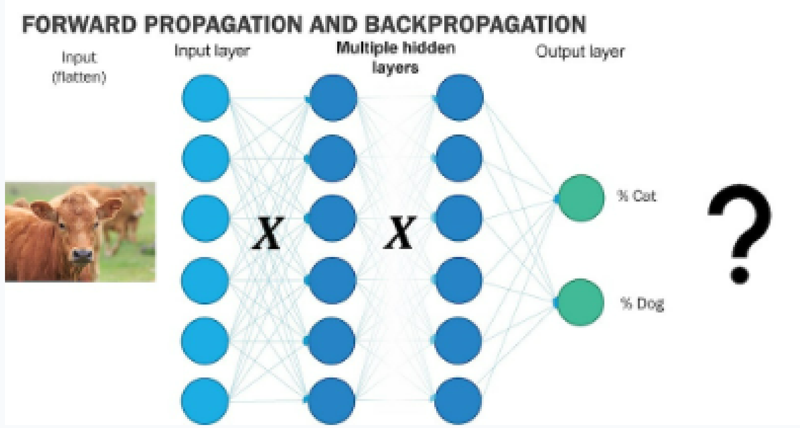
Visual settings

Edit

pollev.com/esterbonmati

When poll is active respond at PollEv.com/esterbonmati

What will the NN predict?



Will predict cow

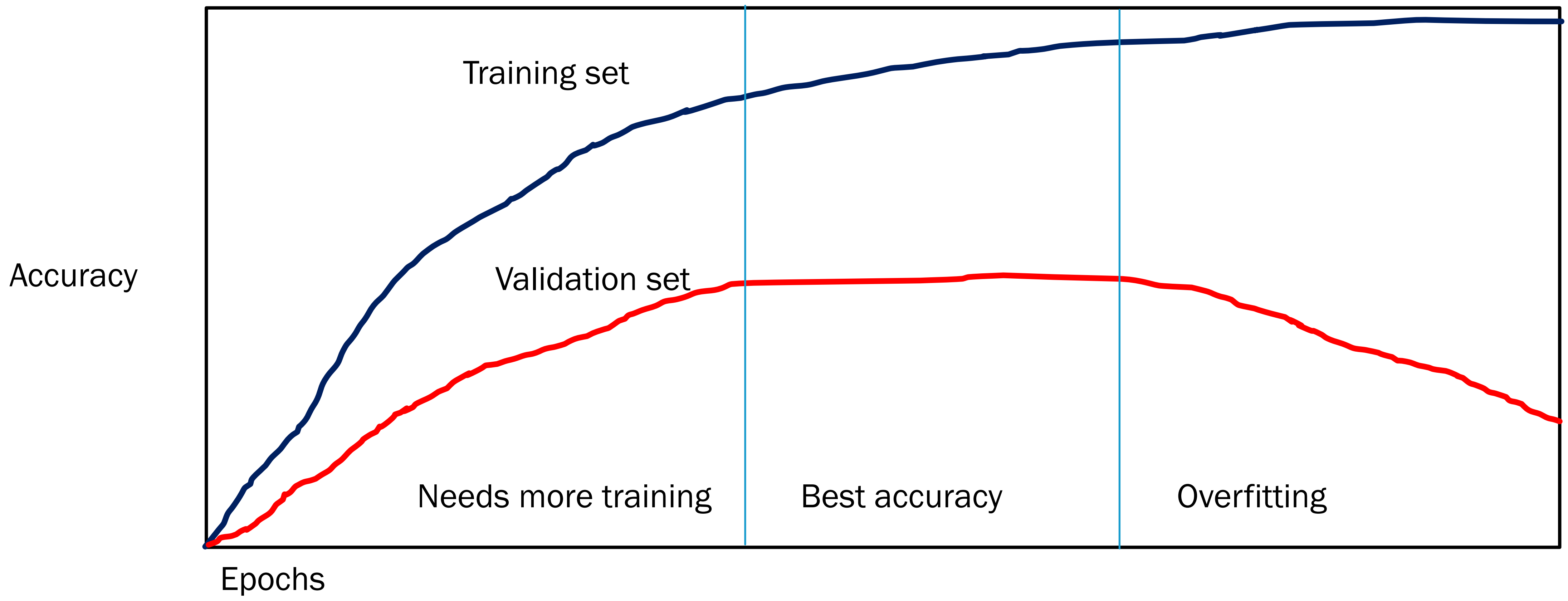
0%

SEE MORE

dict cat or dog

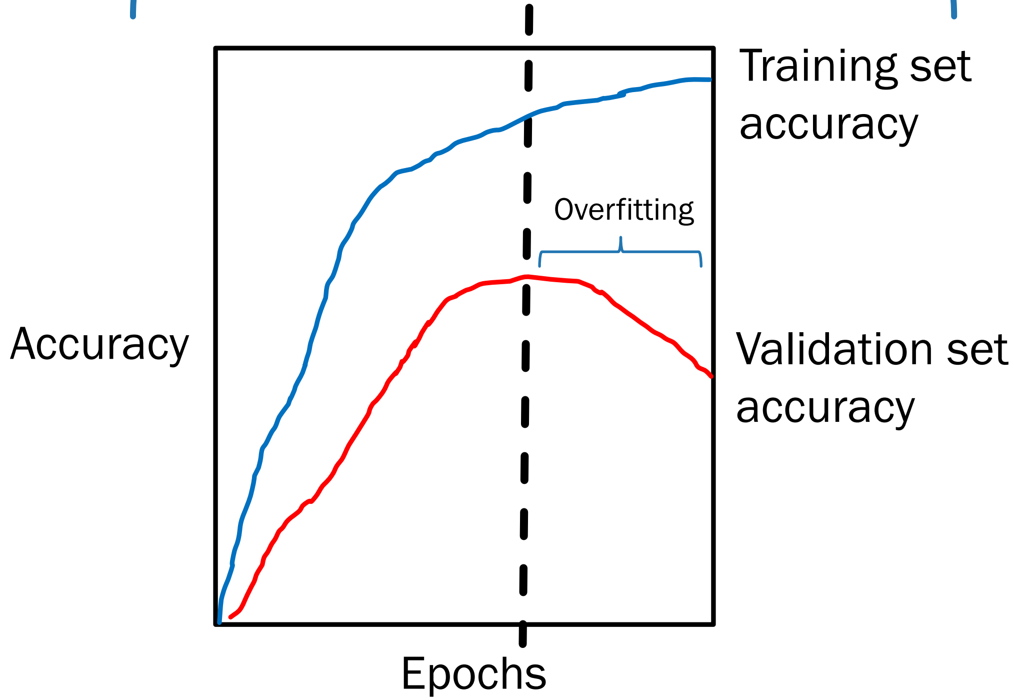
TRAINING

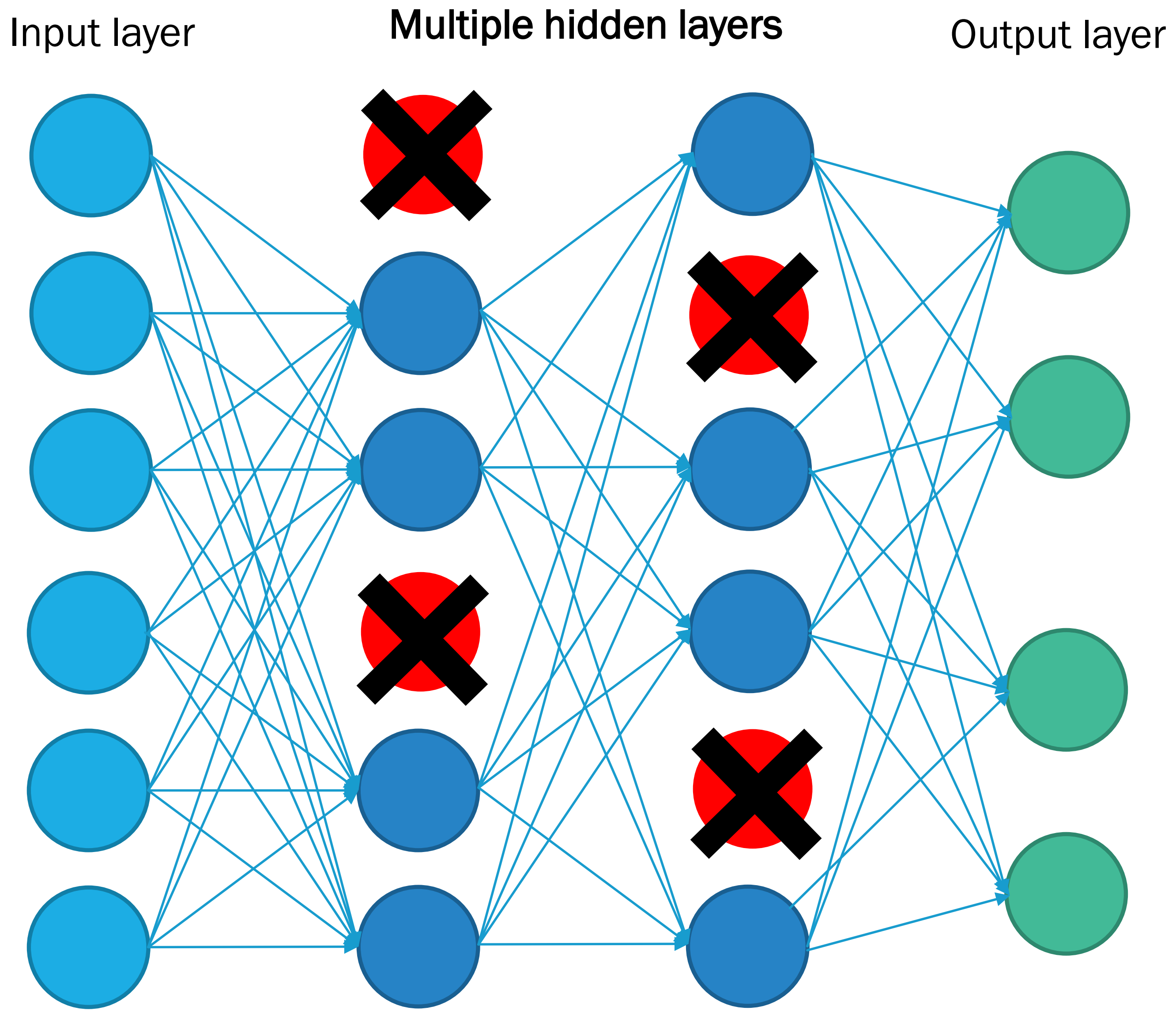
Train model



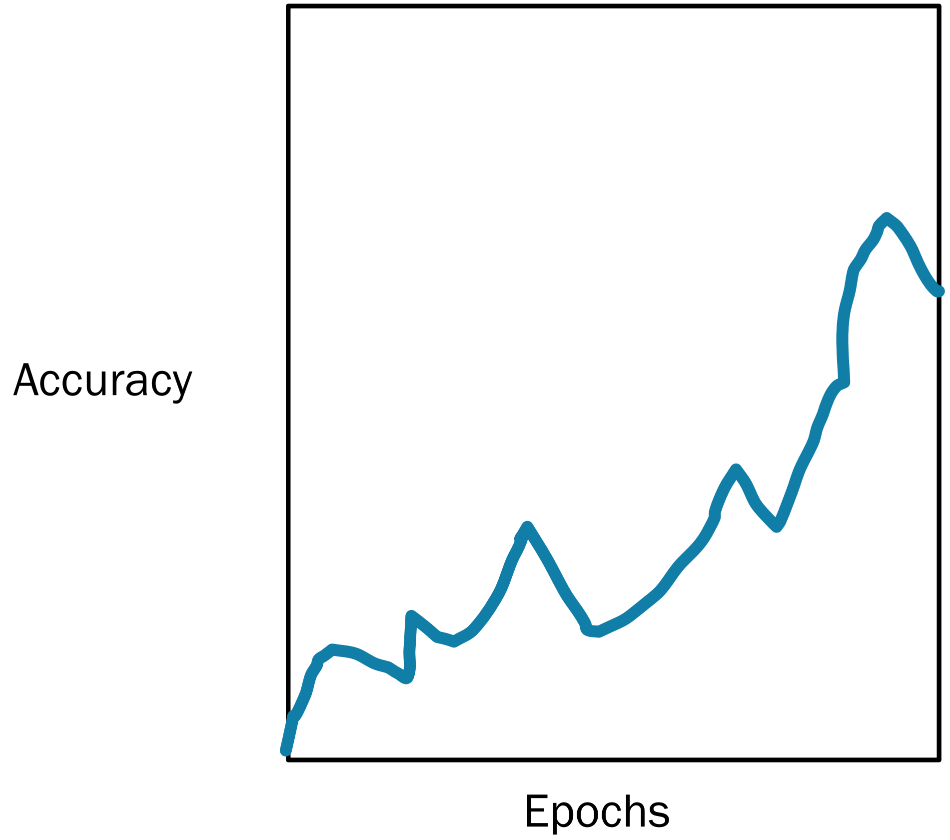
PROBLEMS: OVERFITTING

- Regularisation
  - **Dropout:** During training, we set some activation to 0 (usually 50%)
  - **Early stopping:** stop training before it overfits the data





TRAINING



Applied AI

When poll is active respond at PollEv.com/esterbonmati

What would you do next?

Decrease the learning rate

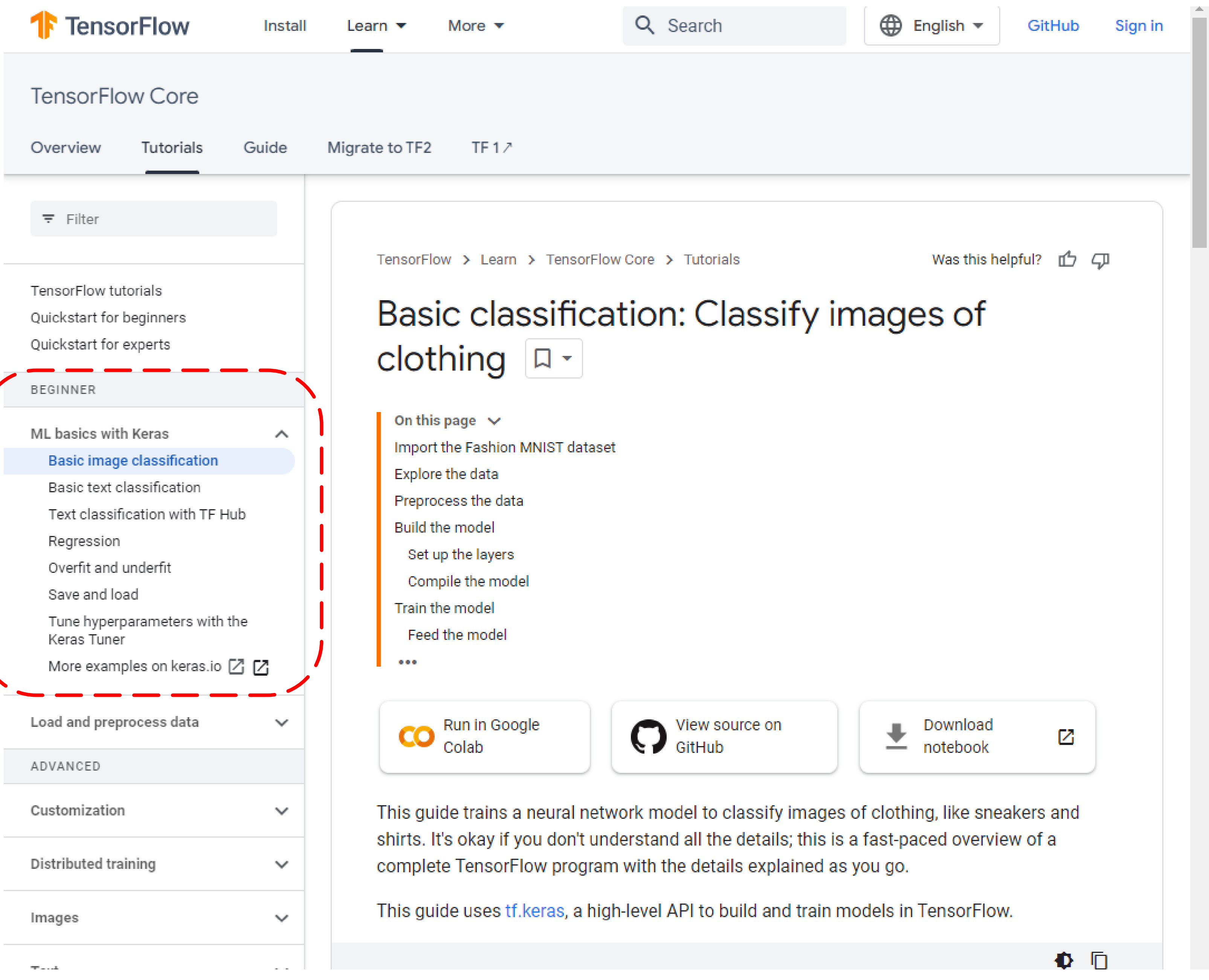
0%

Increase the learning rate

0%

TENSORFLOW & KERAS

- https://www.tensorflow.org/tutorials/keras/classification



PYTORCH

- https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html

PyTorch

Table of Contents

Run in Microsoft Learn | Run in Google Colab | Download Notebook | View on GitHub

Learn the Basics || Quickstart || Tensors || Datasets & DataLoaders || Transforms || **Build Model** || Autograd || Optimization || Save & Load Model

BUILD THE NEURAL NETWORK

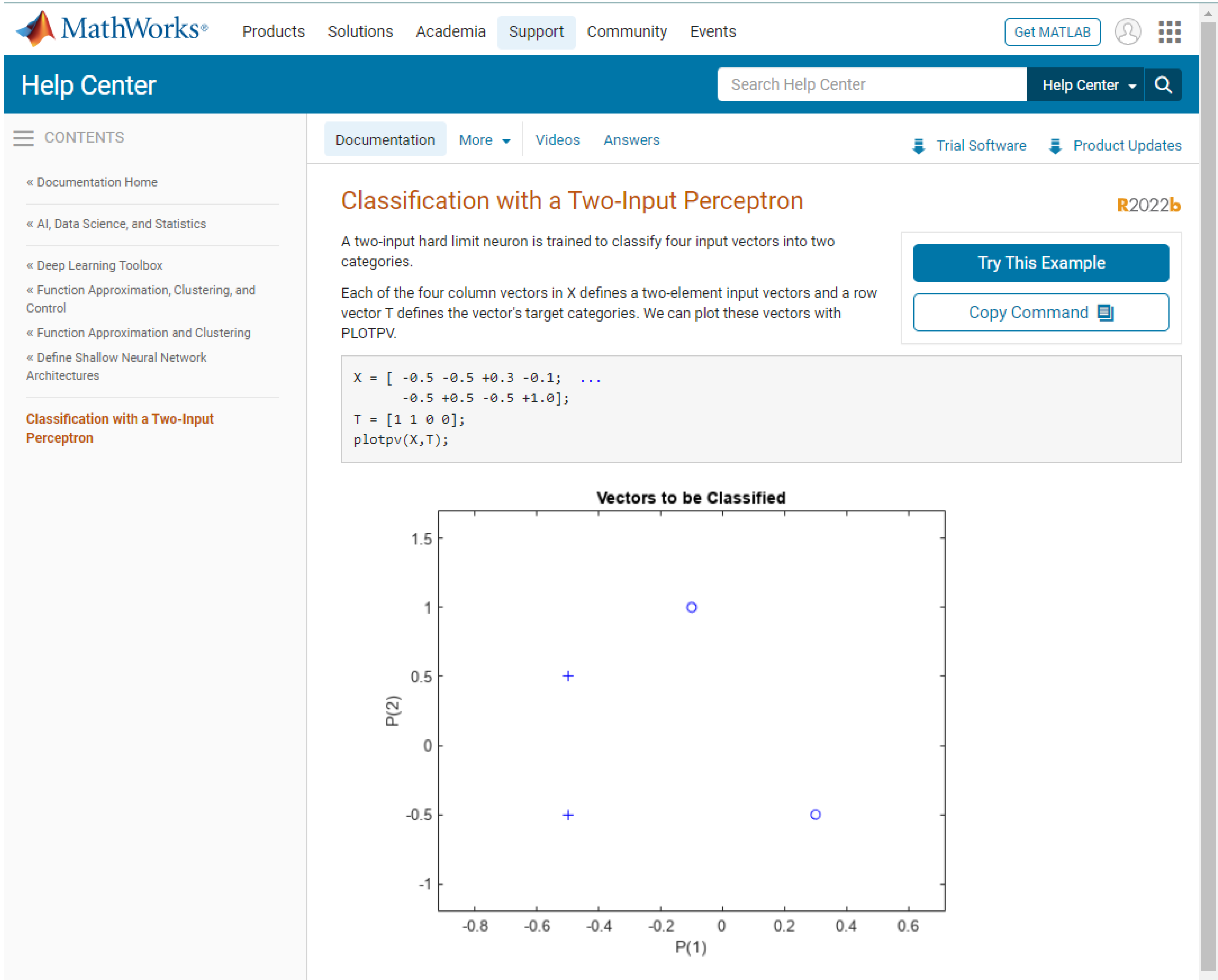
Neural networks comprise of layers/modules that perform operations on data. The torch.nn namespace provides all the building blocks you need to build your own neural network. Every module in PyTorch subclasses the nn.Module. A neural network is a module itself that consists of other modules (layers). This nested structure allows for building and managing complex architectures easily.

In the following sections, we'll build a neural network to classify images in the FashionMNIST dataset.

```
import os
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
```

ALSO IN MATLAB

- https://uk.mathworks.com/help/deeplearning/ug/classification-with-a-2-input-perceptron.html



Applied AI

Moderate Visual settings Edit

pollev.com/esterbonmati

When poll is active respond at PollEv.com/esterbonmati

After: what is a neural network or artificial neural network?

Nobody has responded yet.

Hang tight! Responses are coming in.

ACKNOWLEDGEMENTS

Acknowledgements: This lecture was inspired by the following material:

- http://introtodeeplearning.com/: **MIT Introduction to Deep Learning | 6.S191**
- Neural Networks (Applied AI 21/22 – Dr. Artie Basukoski)