Do you recognise the music playing?

6COSC020W: APPLIED AI

WEEK 8: (ARTIFICIAL) NEURAL NETWORKS

DR. ESTER BONMATI

(e.bonmaticoll@westminster.ac.uk)

LEARNING OUTCOMES OF THIS SESSION

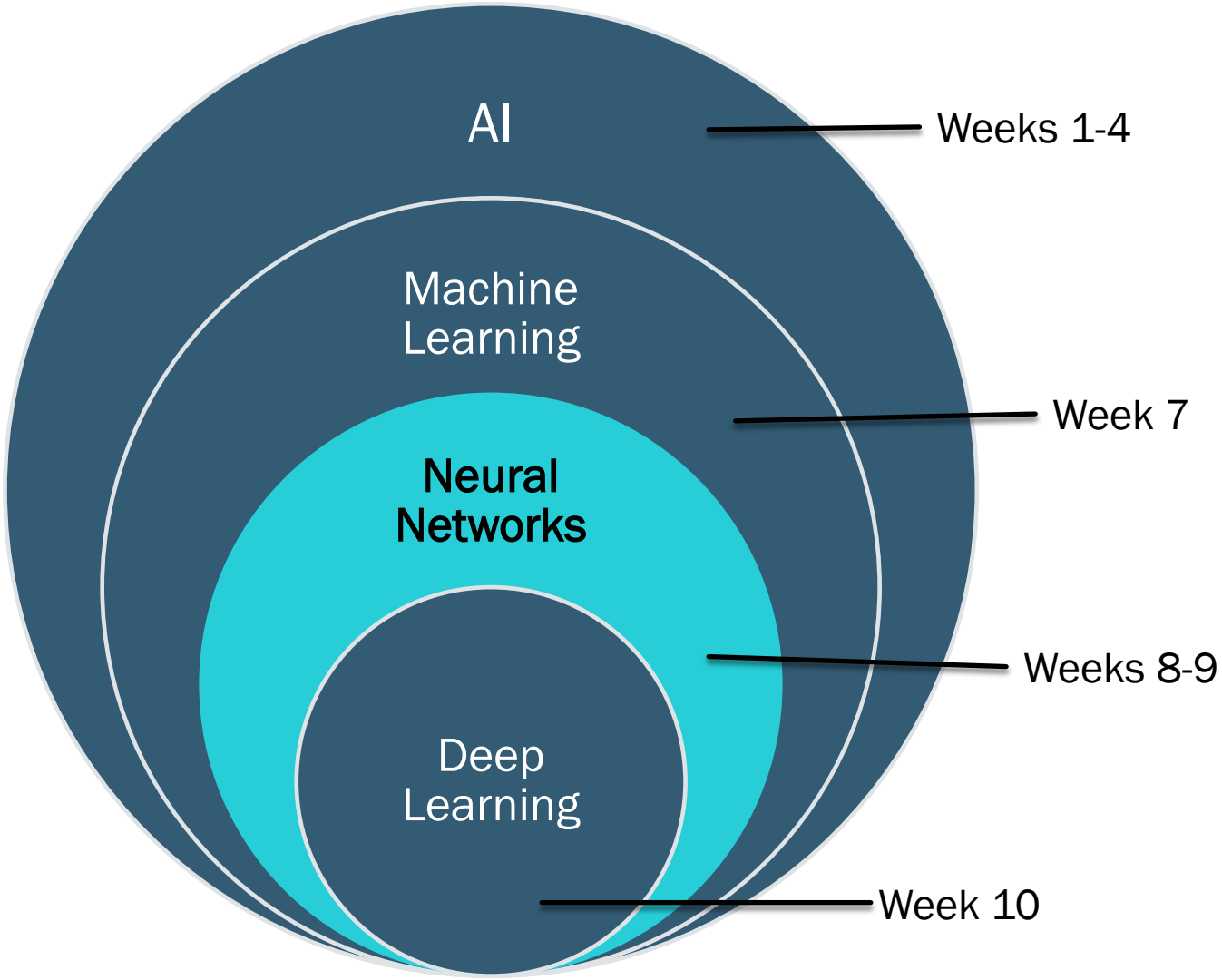
Perceptron

- Understand the fundamental concept
- Understand inputs and outputs
- Understand weights
- Forward propagation
- Understand activation functions
- Be able to calculate the output of a perceptron

Neural Networks

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

NEURAL NETWORKS AND ARTIFICIAL INTELLIGENCE



Activities

Moderate

Visual settings

Edit

When poll is active, respond at **PollEv.com/esterbonmati**

What is a neural network?

No responses received yet. They will appear here...

Powered by Poll Everywhere

ARTIFICIAL NEURAL NETWORKS (ANN)

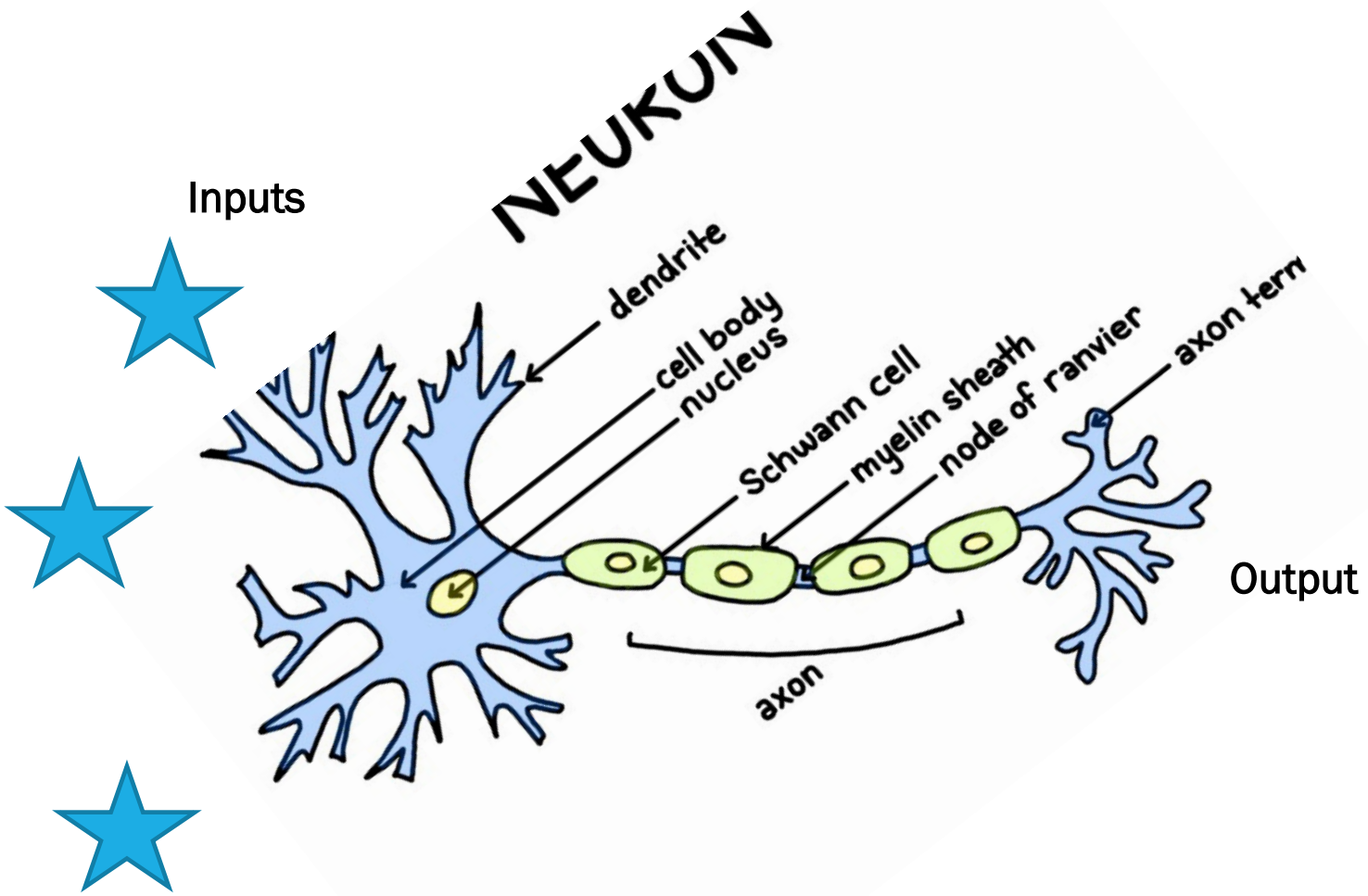
Why are they important?

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.

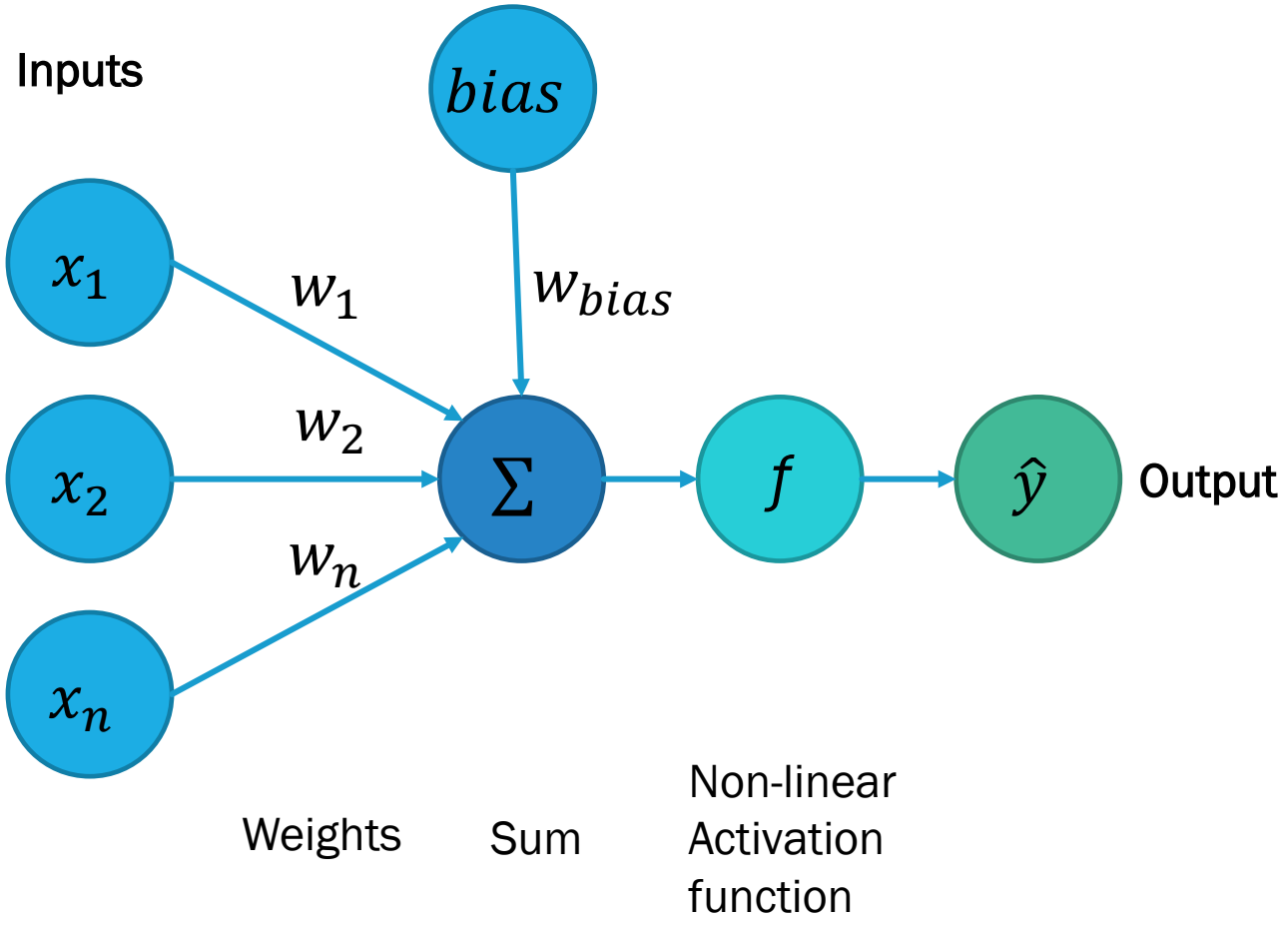
WHAT DO THEY ALL HAVE IN COMMON?

WHAT DO THEY ALL HAVE IN COMMON?

NEURON

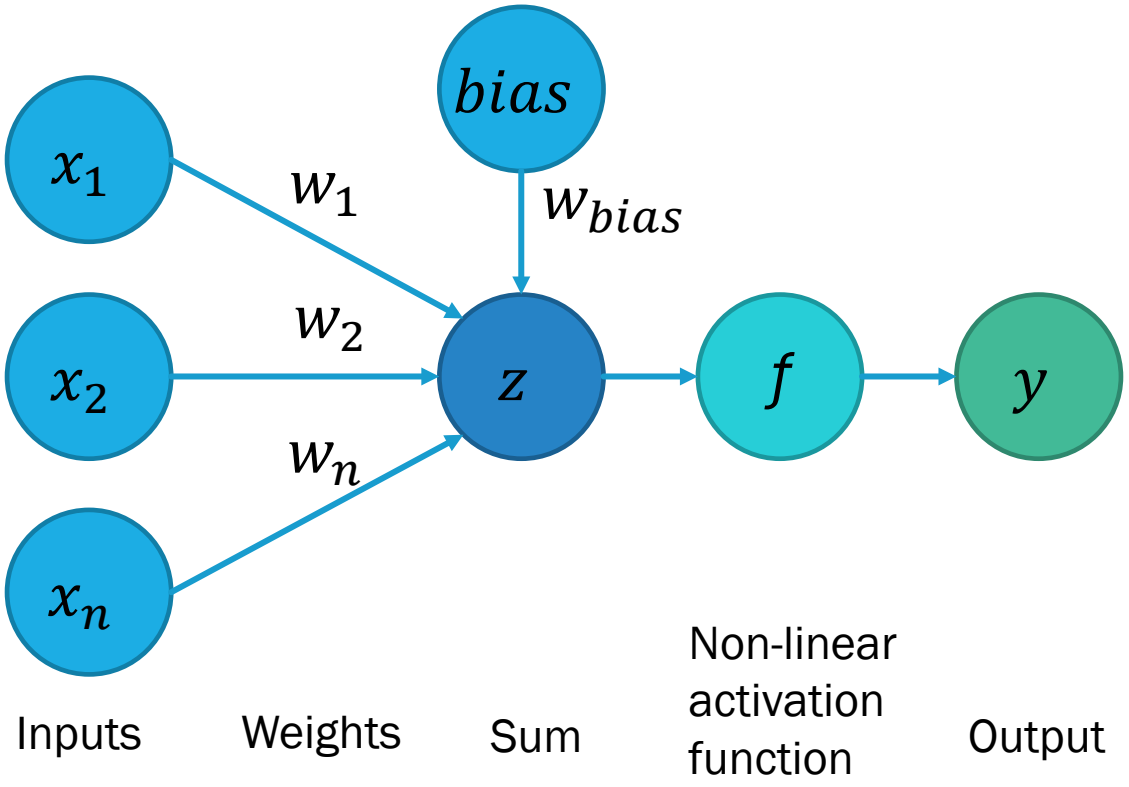

Inputs
NEURON
dendrite
cell body
nucleus
Schwann cell
myelin sheath
node of ranvier
axon terr
Output
axon

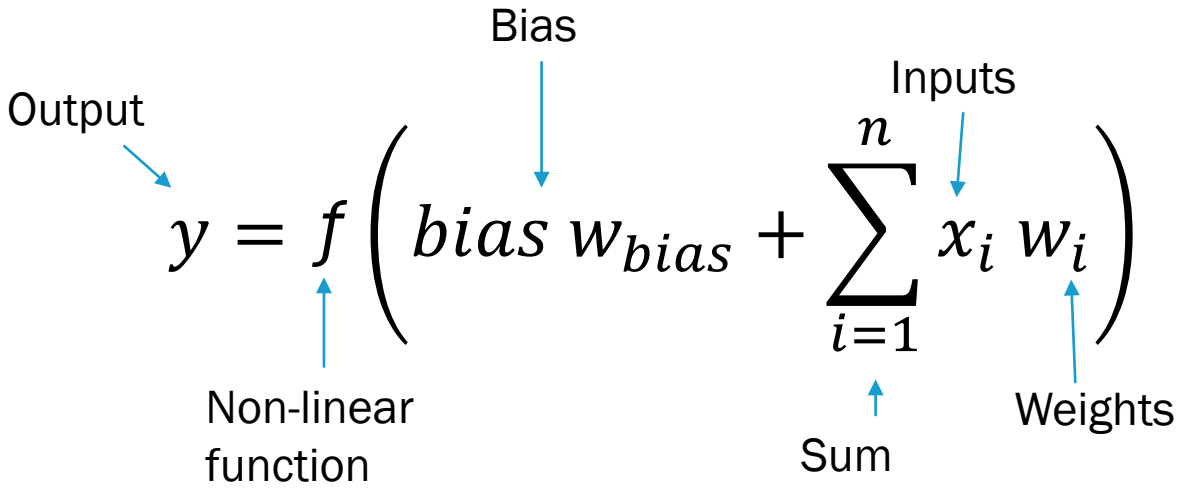
THE PERCEPTRON



THE PERCEPTRON

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



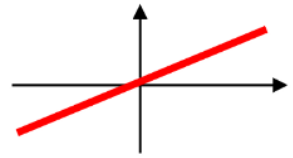
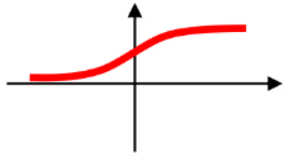
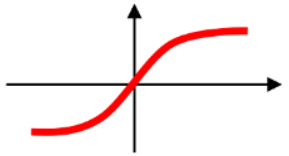
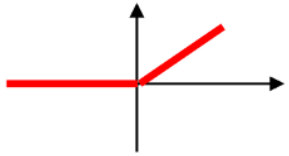


$$z = x_1\ w_1 + x_2\ w_2 + \ldots + x_n\ w_n + bias\ w_{bias}$$

$$y = f(Z)$$

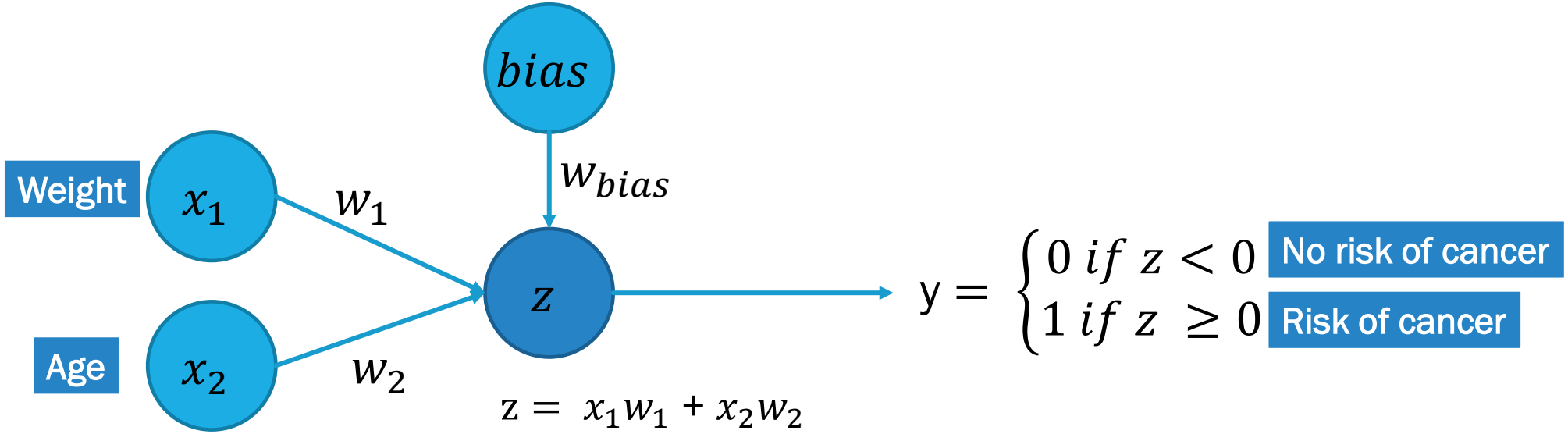
Example: $y = \begin{cases} 0\ if\ z < 0 \\ 1\ if\ z \geq 0 \end{cases}$

FOR REFERENCE: 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

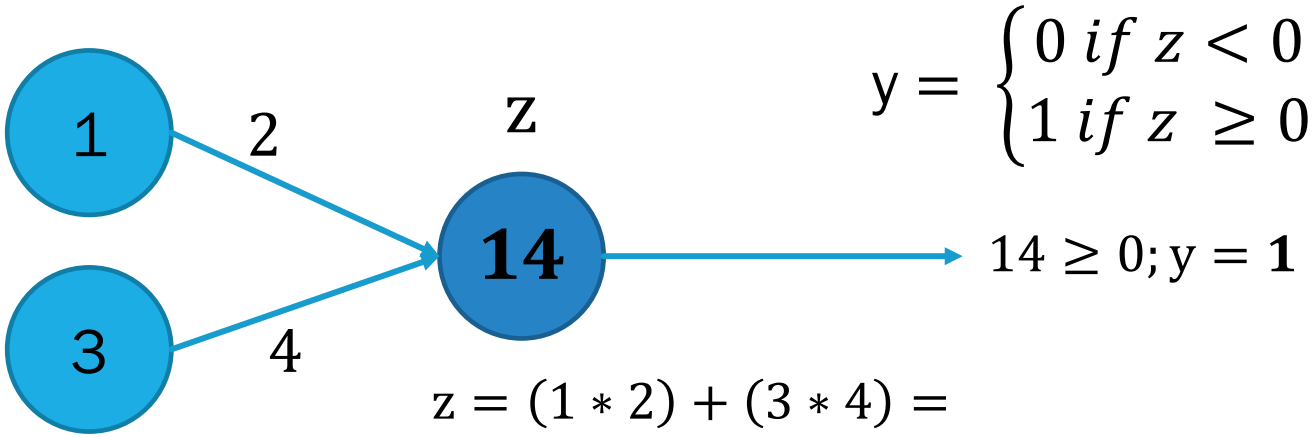
PERCEPTRON: FORWARD PROPAGATION



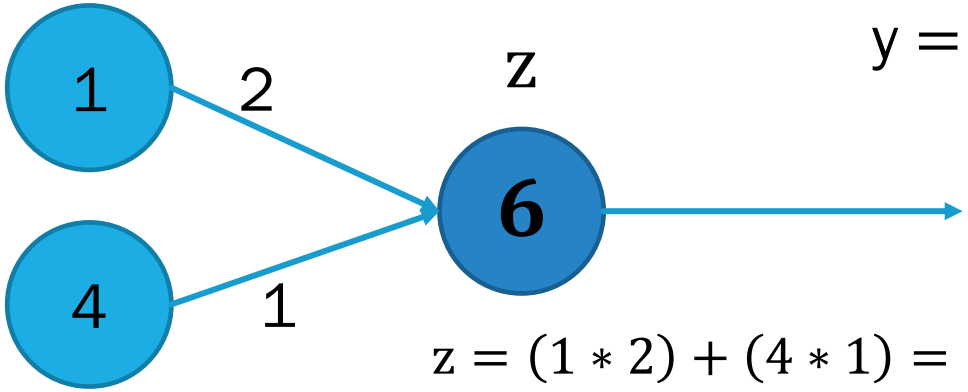
Do you know another example of a perceptron?

- Logic gates
- Predict who is going to pass this module based on the number of lectures and tutorials attended

PERCEPTRON: FORWARD PROPAGATION EXAMPLE



PERCEPTRON: FORWARD PROPAGATION EXAMPLE



$$y = \begin{cases} 0 \; if \; z < 10 \\ 1 \; if \; z \geq 10 \end{cases}$$

$6 < 10; y = \mathbf{0}$

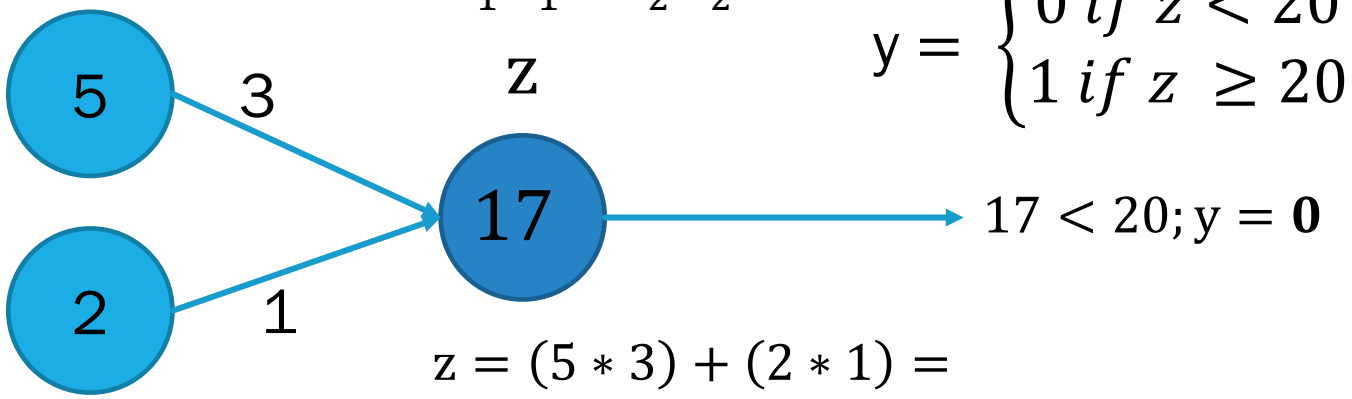
$z = (1 * 2) + (4 * 1) =$
$= 2 + 4 = \mathbf{6}$

Do you think you can do it?

PERCEPTRON: FEEDBACK

$$z = x_1 w_1 + x_2 w_2$$

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

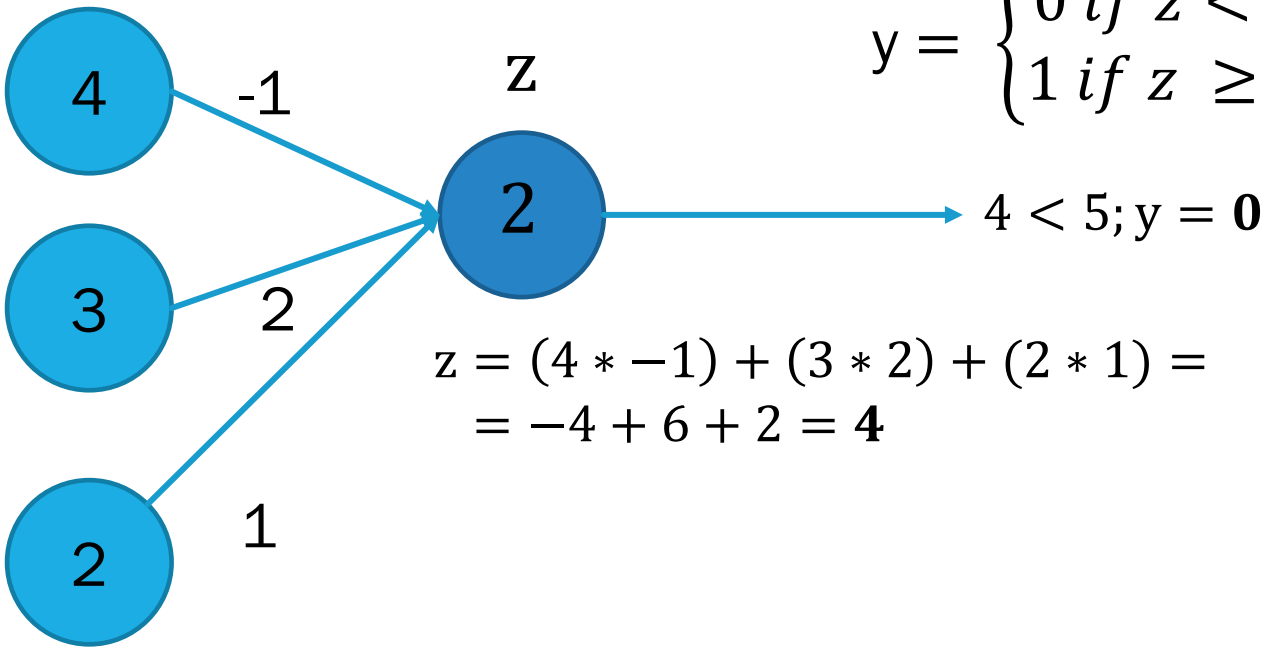


$17 < 20; y = \mathbf{0}$

$$z = (5 * 3) + (2 * 1) =$$
$$= 15 + 2 = \mathbf{17}$$

Compare answers with the person sitting next to you

PERCEPTRON: FEEDBACK



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

$4 < 5; y = \mathbf{0}$

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

Compare answers with the person sitting next to you

Activities

Results perceptron 2

Join by Web

1 Go to **PollEv.com**

2 Enter **ESTERBONMATI**

3 Respond to activity

Total Results: 0

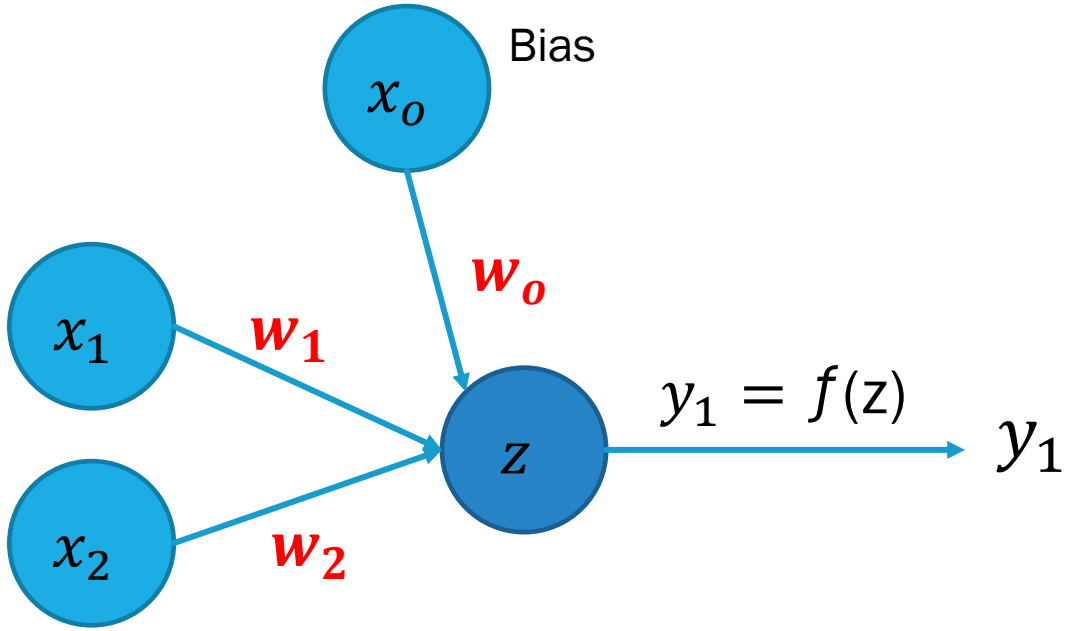
Powered by Poll Everywhere

Activities

When poll is active, respond at **PollEv.com/esterbonmati**

How confident do you feel in calculating the output of a perceptron?

PERCEPTRON: A PRACTICAL EXAMPLE WITH ONLY 2 INPUTS (PYTHON)

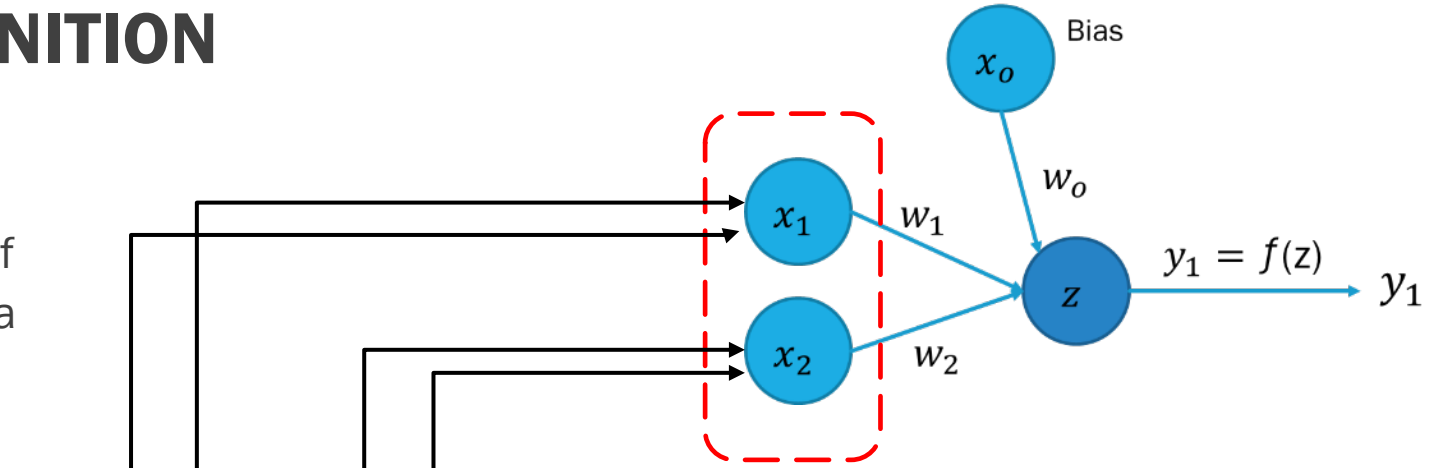


What do the weights mean and how do we find them?

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

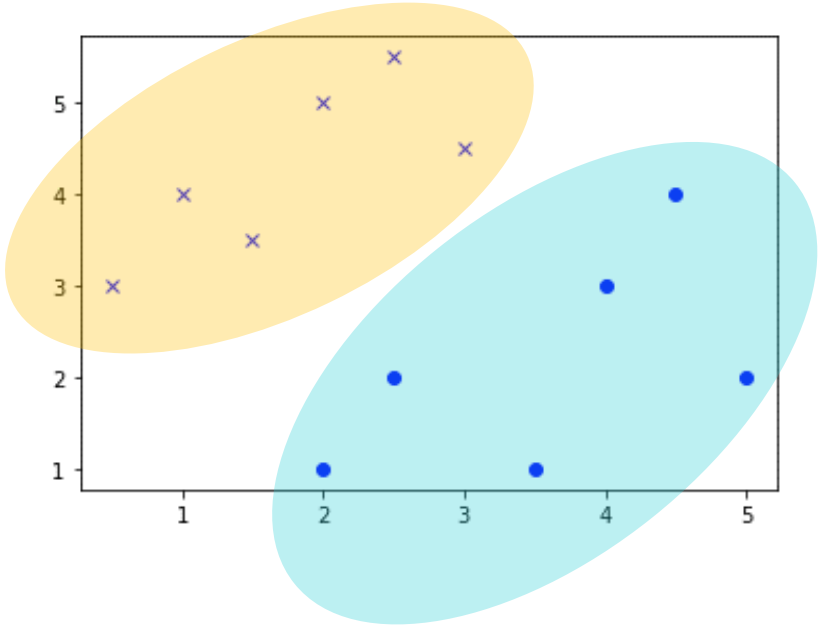
PERCEPTRON: INPUT DEFINITION

Problem: 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.



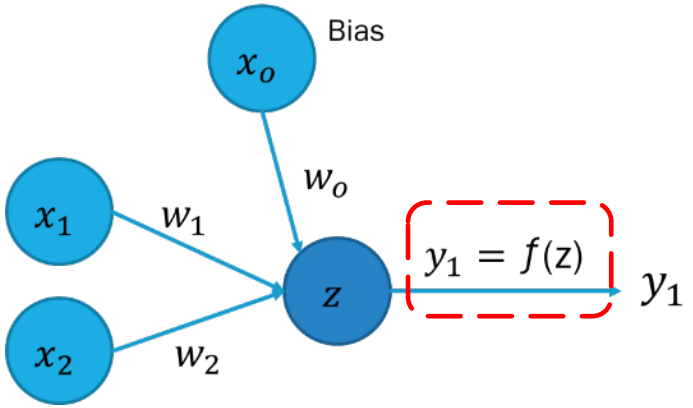

```
x1_group1 = [2, 5, 4, 2.5, 3.5, 4.5]
x2_group1 = [1, 2, 3, 2, 1, 4]

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

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



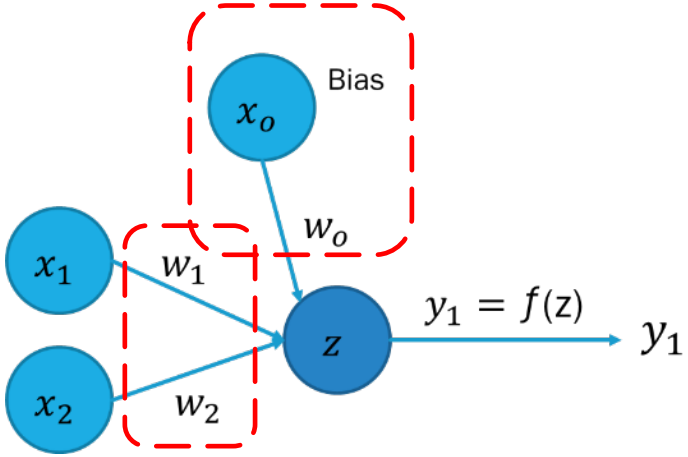
If the sum is is greater than 0 we return 1, otherwise we return 0

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

PERCEPTRON: WEIGHTS AND BIAS DEFINITION

Some random weights…

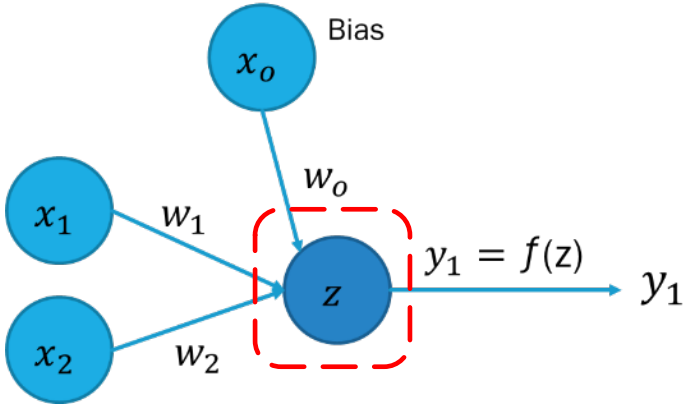
```
w1 = 0.75
w2 = -0.5
bias = 1
wbias = 1
```



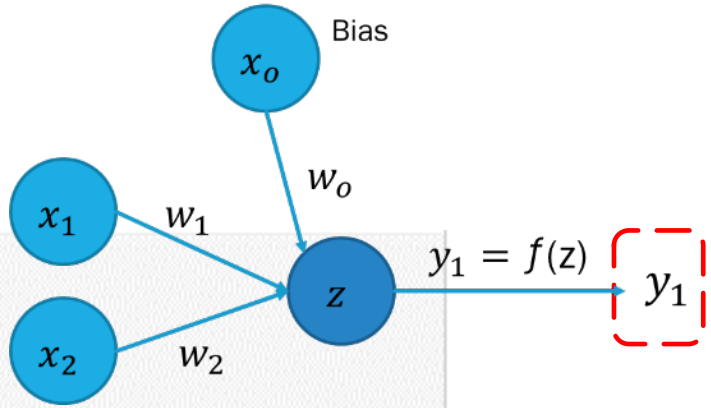
How good are these weights in grouping the input data?

PERCEPTRON: FORWARD PROPAGATION

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



PERCEPTRON: PREDICTION AND RESULTS




```
    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
```

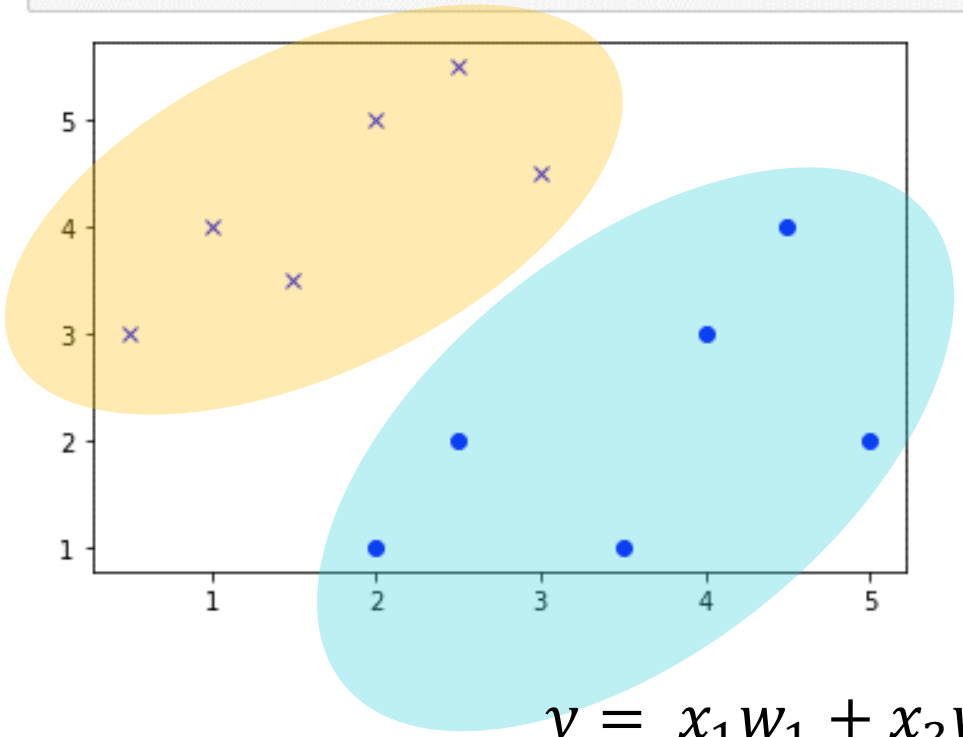
$$accuracy = \frac{number\ correct\ samples}{total\ number\ of\ samples} = \frac{9}{12} = 0.75 \rightarrow \mathbf{75}\ \%$$

Can we improve the accuracy if we change the weights?

PERCEPTRON: VISUALISATION

Plot the points

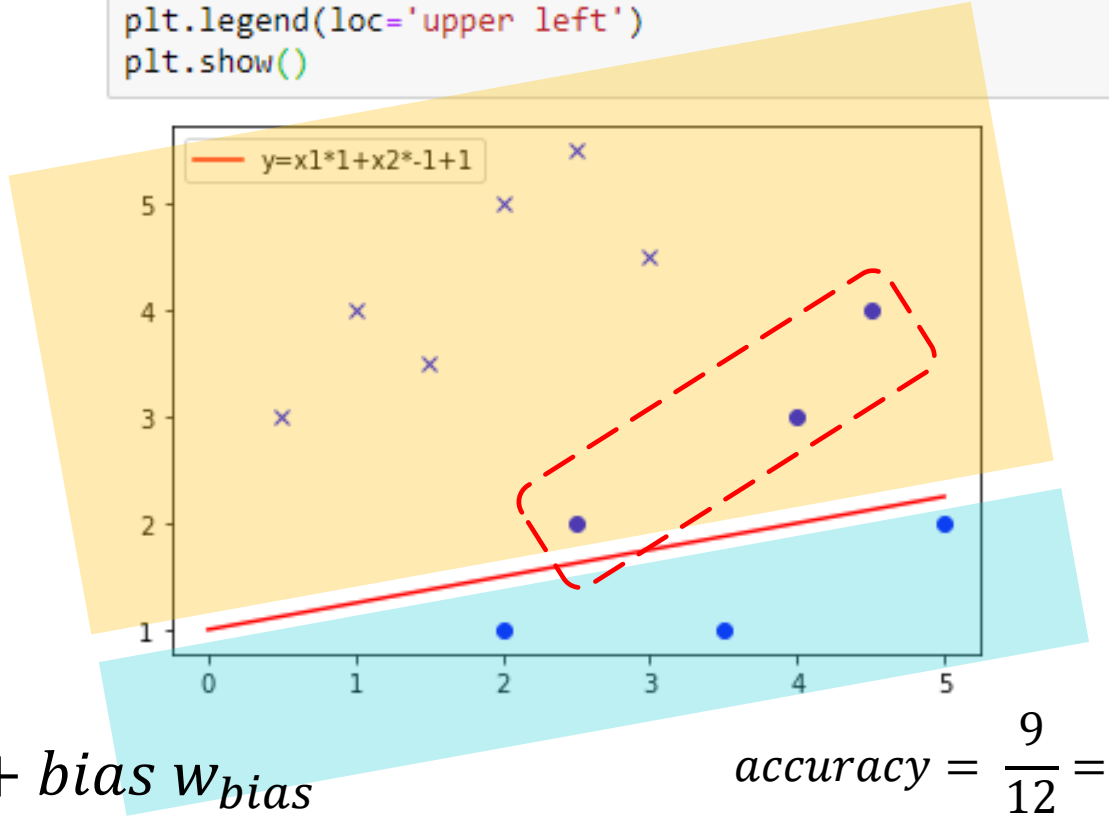
```
In [117]: fig = plt.figure()
          ax = plt.axes()
          plt.plot(x1_group1, x2_group1, 'ob')
          plt.plot(x1_group0, x2_group0, 'xb')
          plt.show()
```



Plot linear activation

```
In [101]: 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()
```



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

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

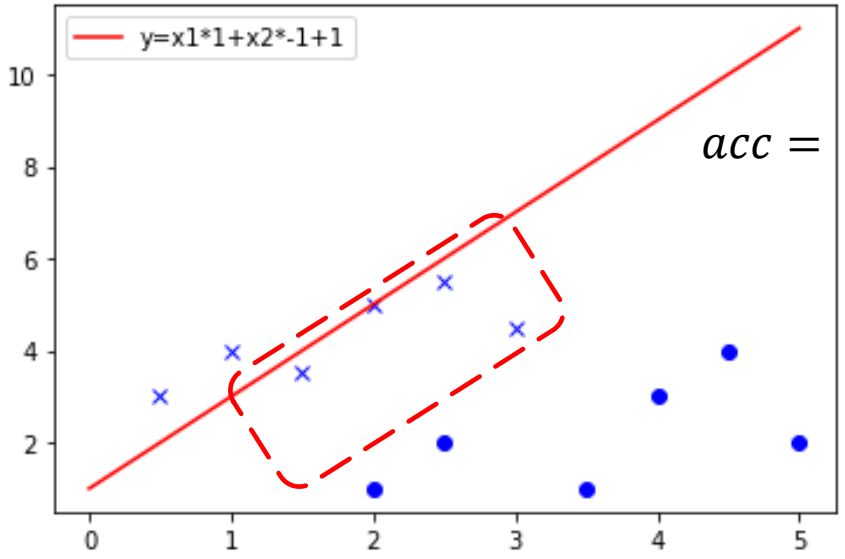
PERCEPTRON: UPDATE WEIGHTS

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()
```

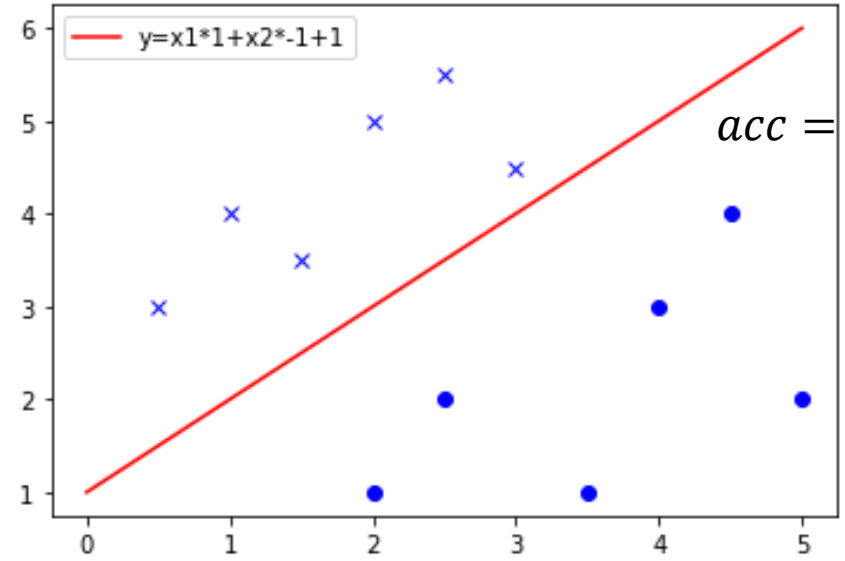


$$acc = \frac{8}{12} = 0.67$$

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

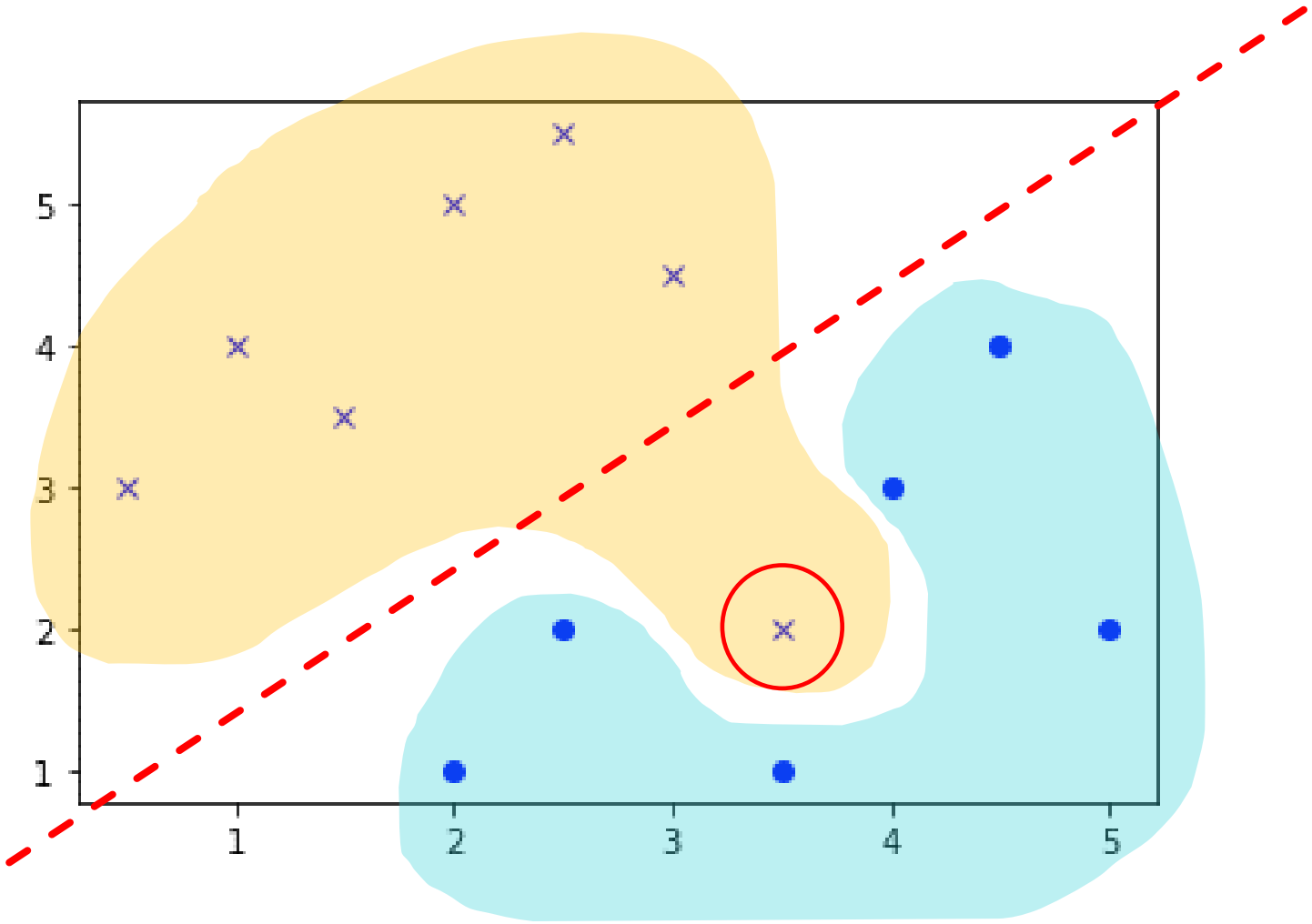
In [104]:
```
for i in range(len(x1_group1)):
    sum = perceptron(x1_group1[i], x2_group1[i], bias)
    group = activation_function(sum)
    print('Point ' + str(i) + ' in group 1 returned ' + str(group))

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
```

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

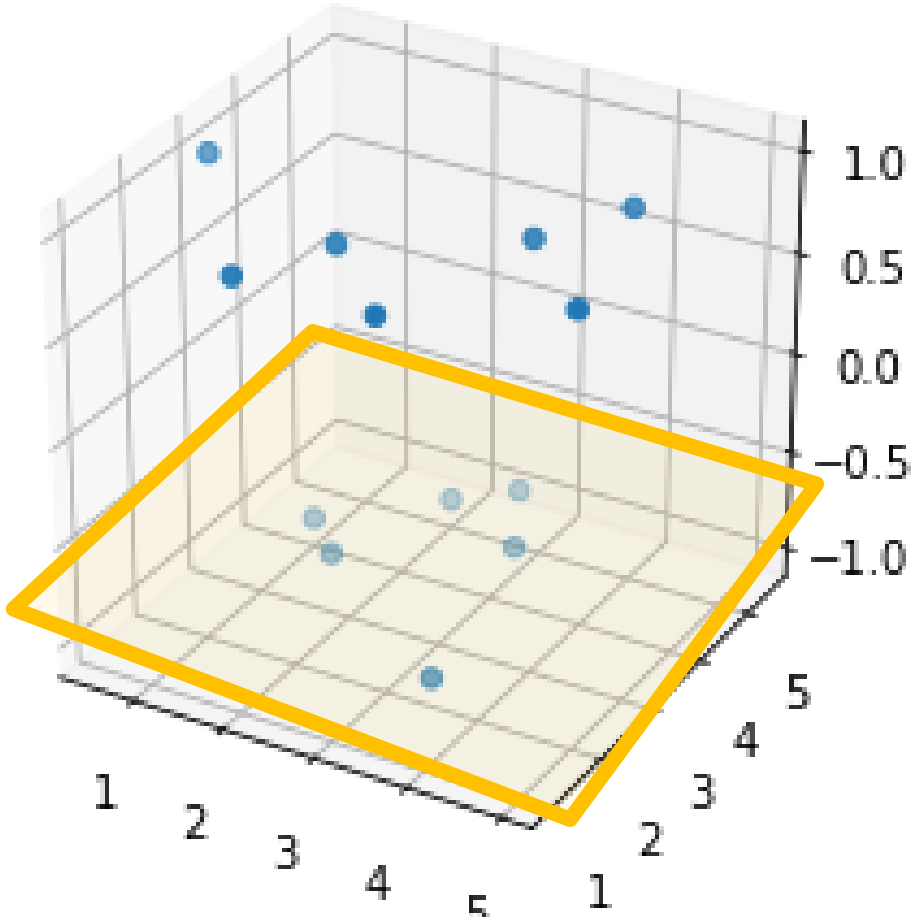
NEURAL NETWORKS: ANOTHER EXAMPLE

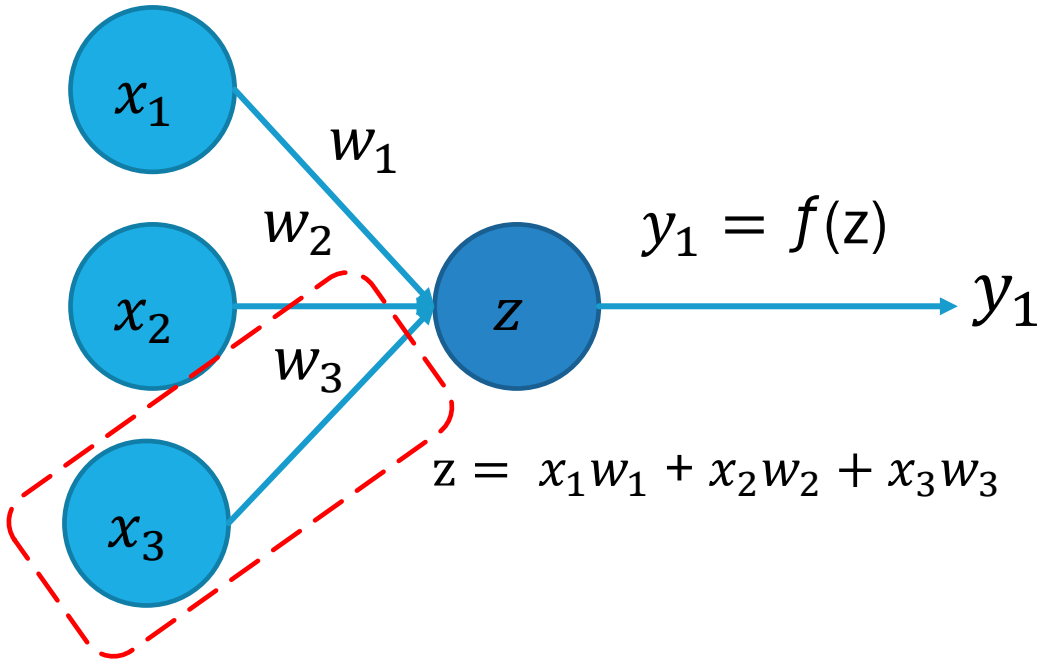


NEURAL NETWORKS: ANOTHER EXAMPLE

Bias

$$y = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$





When poll is active, respond at **PollEv.com/esterbonmati**

How many inputs can have a perceptron?

2

3

n

When poll is active, respond at **PollEv.com/esterbonmati**

How many weights has each input?

0

1

2

n

When poll is active, respond at **PollEv.com/esterbonmati**

We have a perceptron with 2 inputs. How do you calculate Z?

$z = x1 * w1 + x2 * w2 + bias * wbias$

$z = x1 * w1 + x2 * w2$

$z = x1 * w1 + x2 * w2 + x3 * w3 + bias * wbias$

Activities

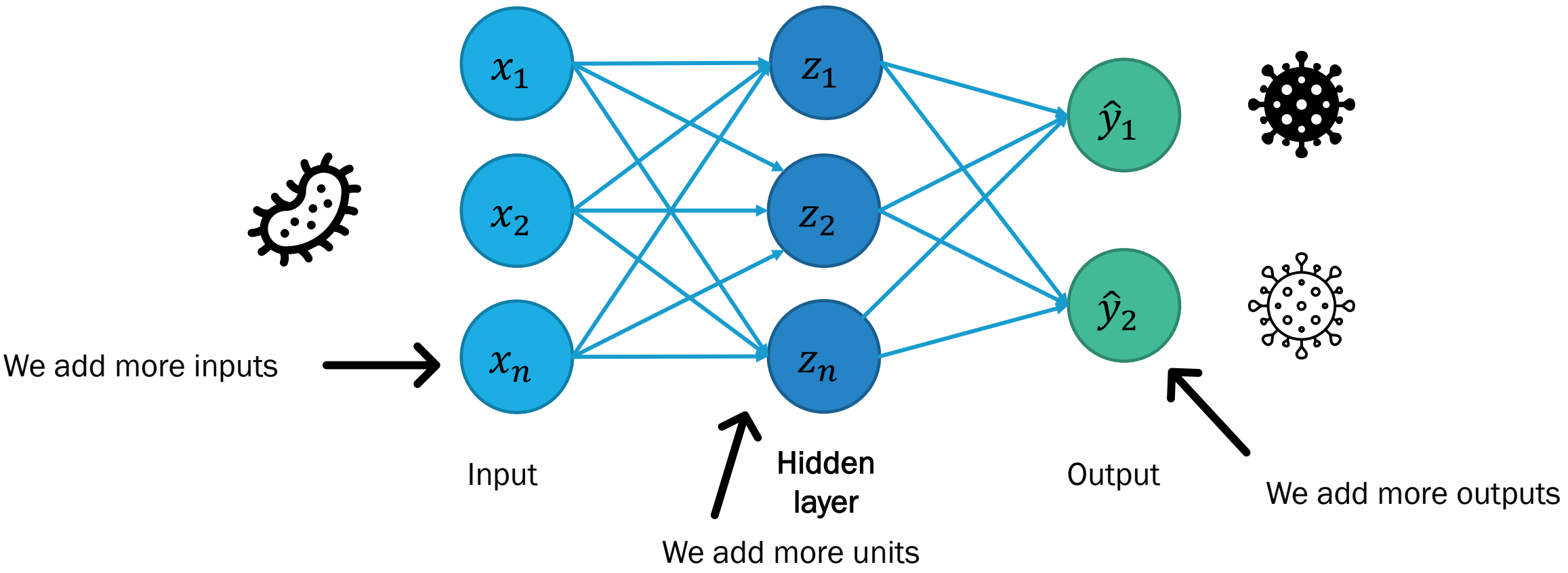
When poll is active, respond at **PollEv.com/esterbonmati**

Q&A

Top

No responses received yet. They will appear here...

FROM PERCEPTRON TO A NEURAL NETWORK



NEURAL NETWORK EXAMPLE

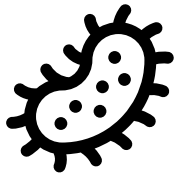
Input

Input layer

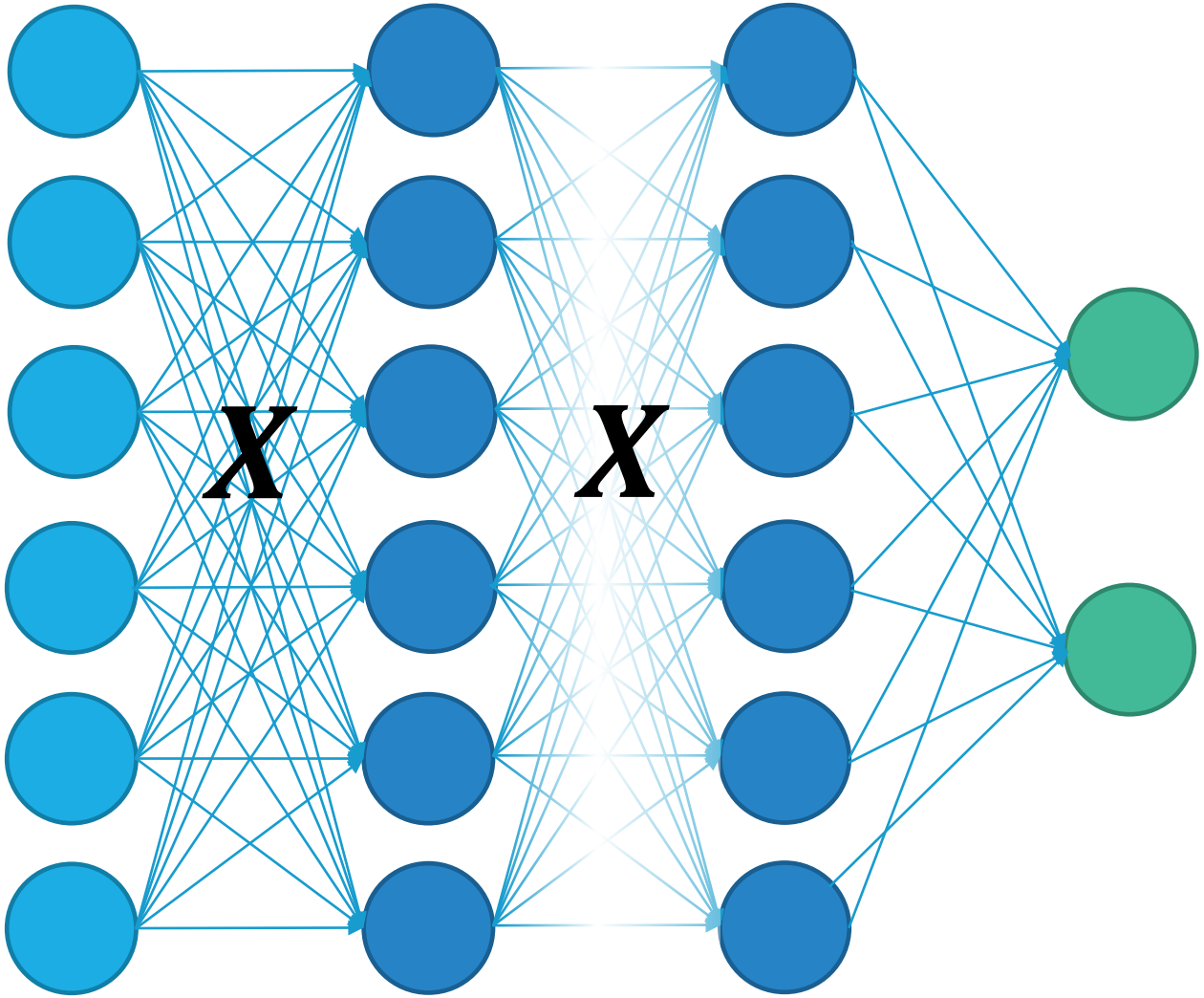
Multiple hidden layers

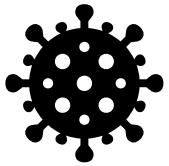
Output layer

Dense layer
==
Fully connected layer



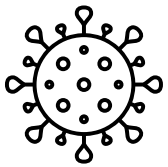
data



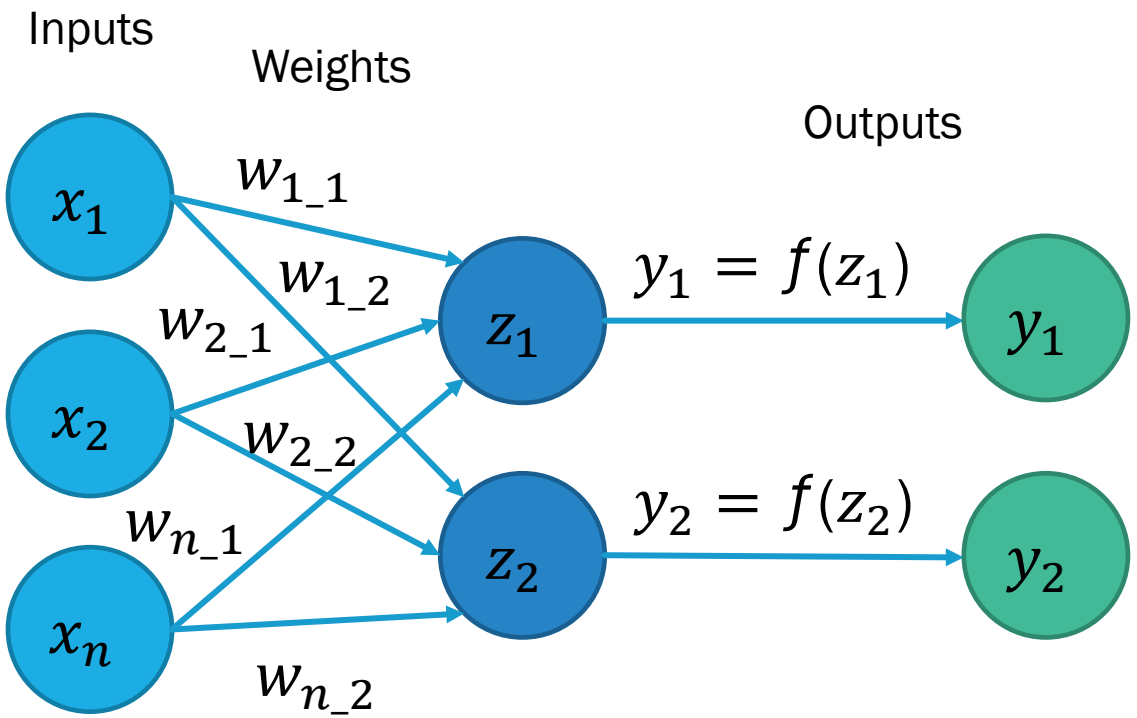


Probability of cancer
95%

Probability of no cancer
5%



ARTIFICIAL NEURAL NETWORKS (ANN)

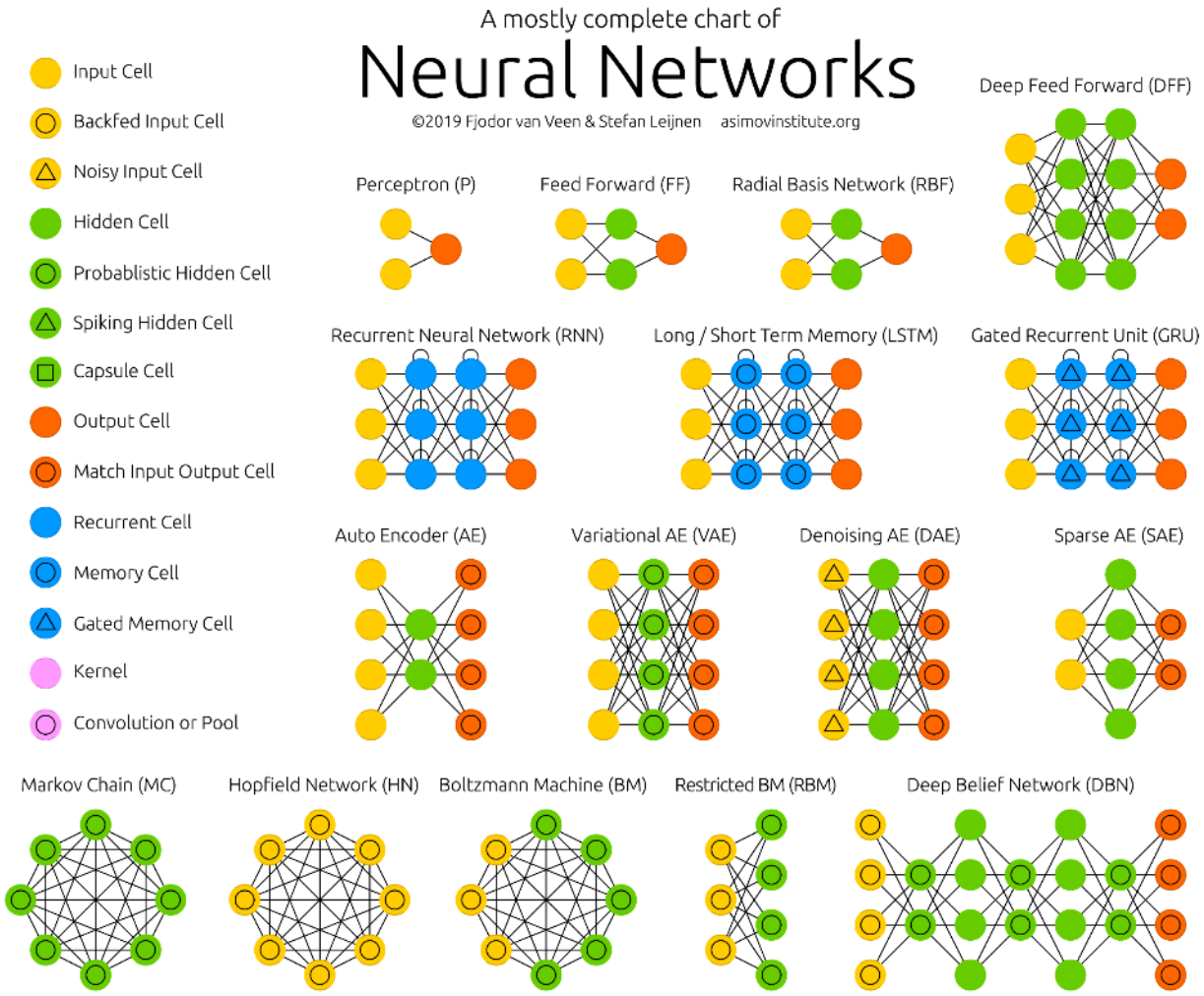


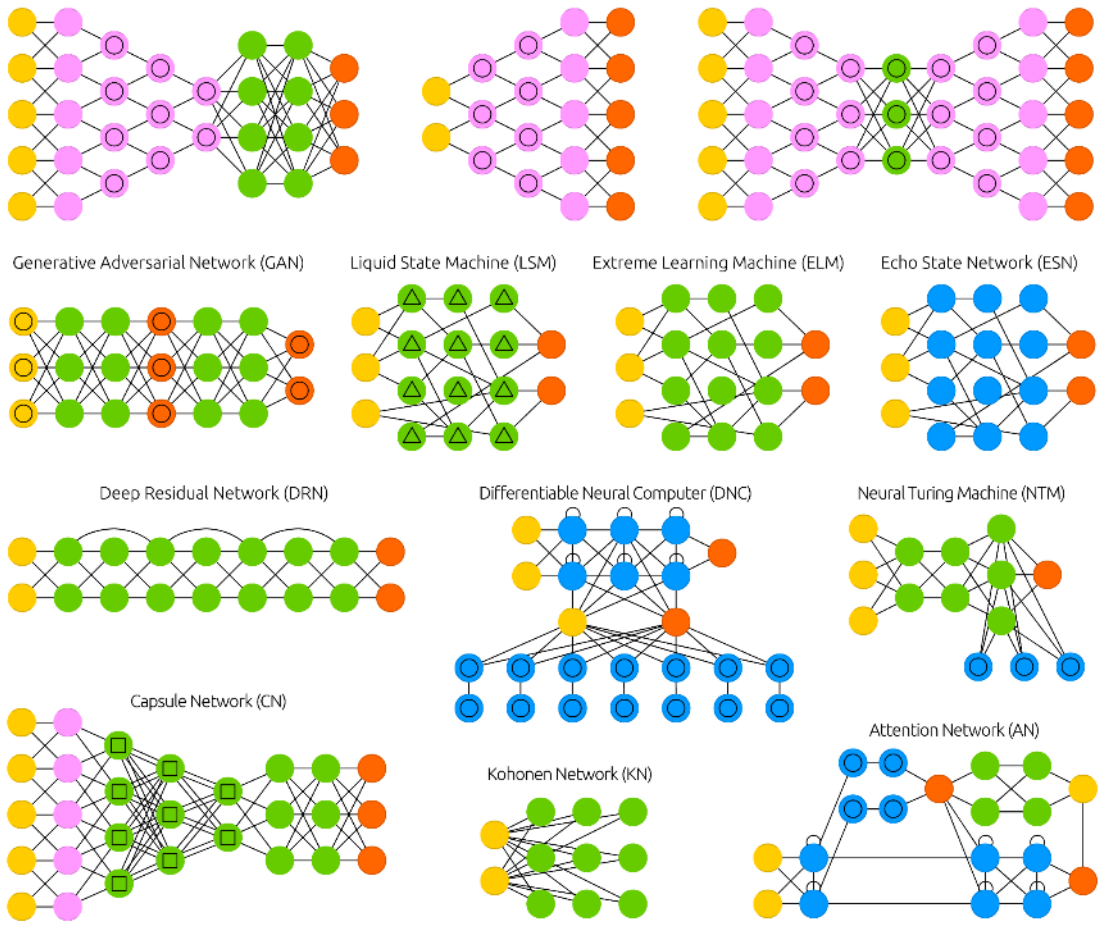
Each output has its own weight for each input

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

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

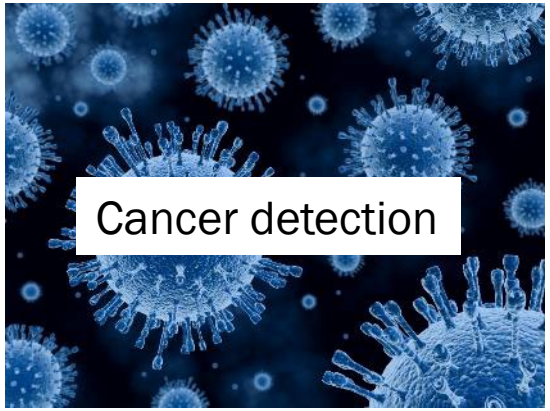
EXAMPLES OF NEURAL NETWORKS





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

WHAT CAN WE DO WITH ANN?



Shopping prediction

Recognition/Classification

Chatbots

Stock market prediction

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

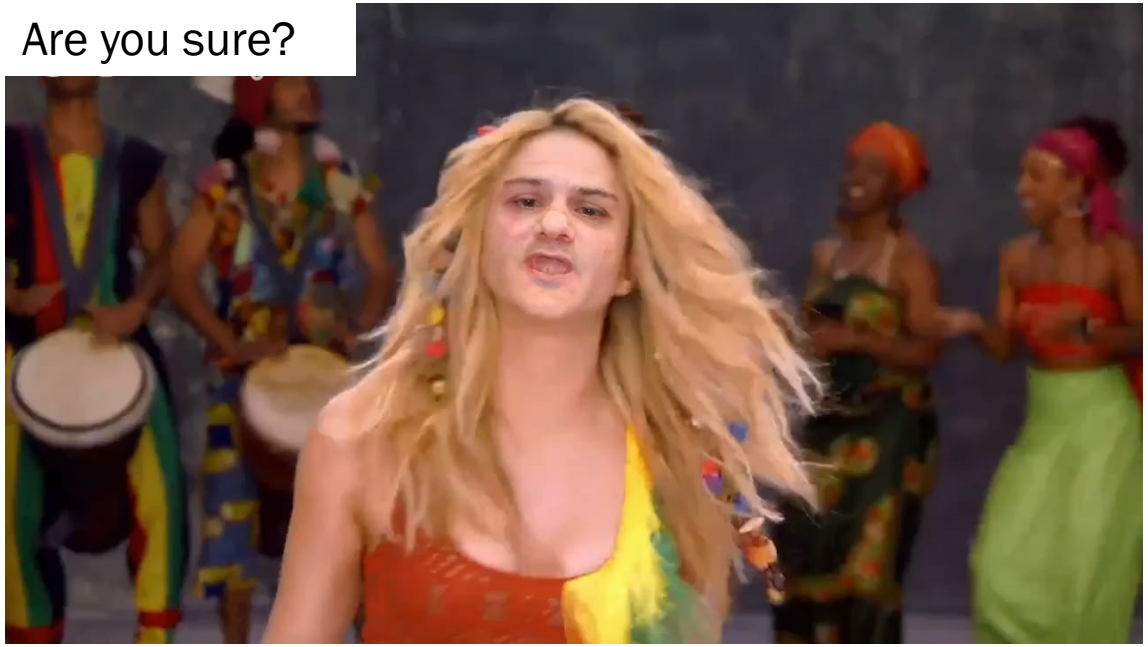
Deep learning in cancer diagnosis, prognosis and treatment selection

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

DEEP FAKE

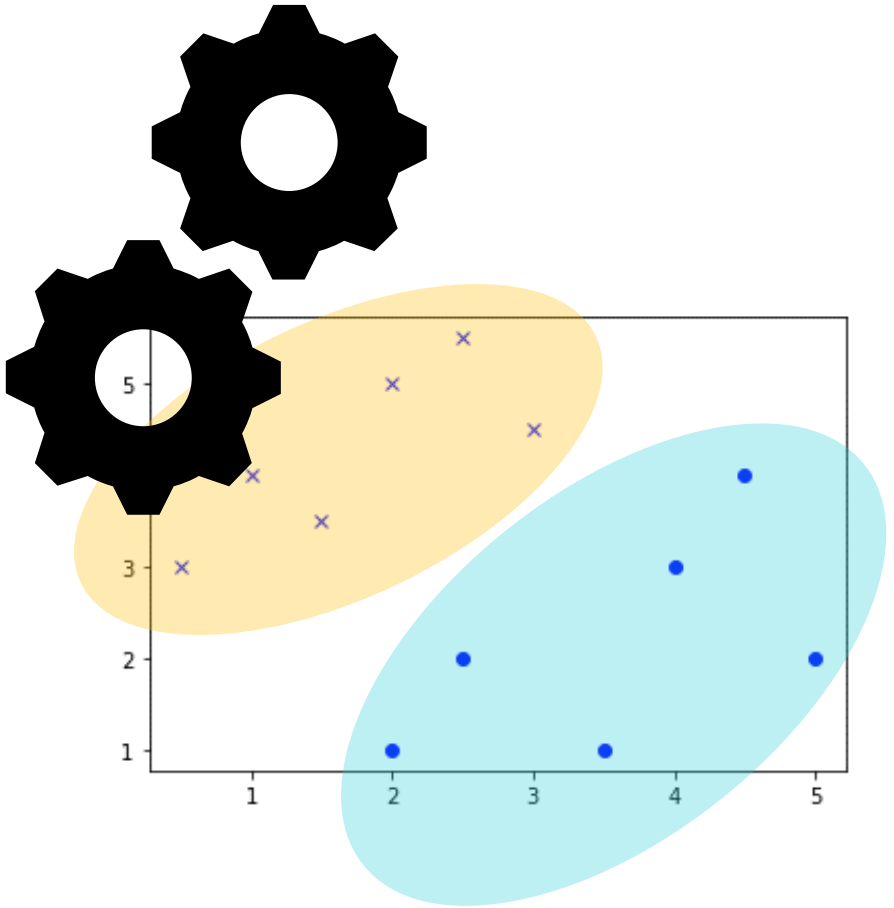
- Synthetic image or video generated using someone else's face or a new generated face.
- Brings concerns about fraud, credibility and authenticity, etc.

Are you sure?



Video generated using FaceMagic APP

SUPERVISED VS UNSUPERVISED



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

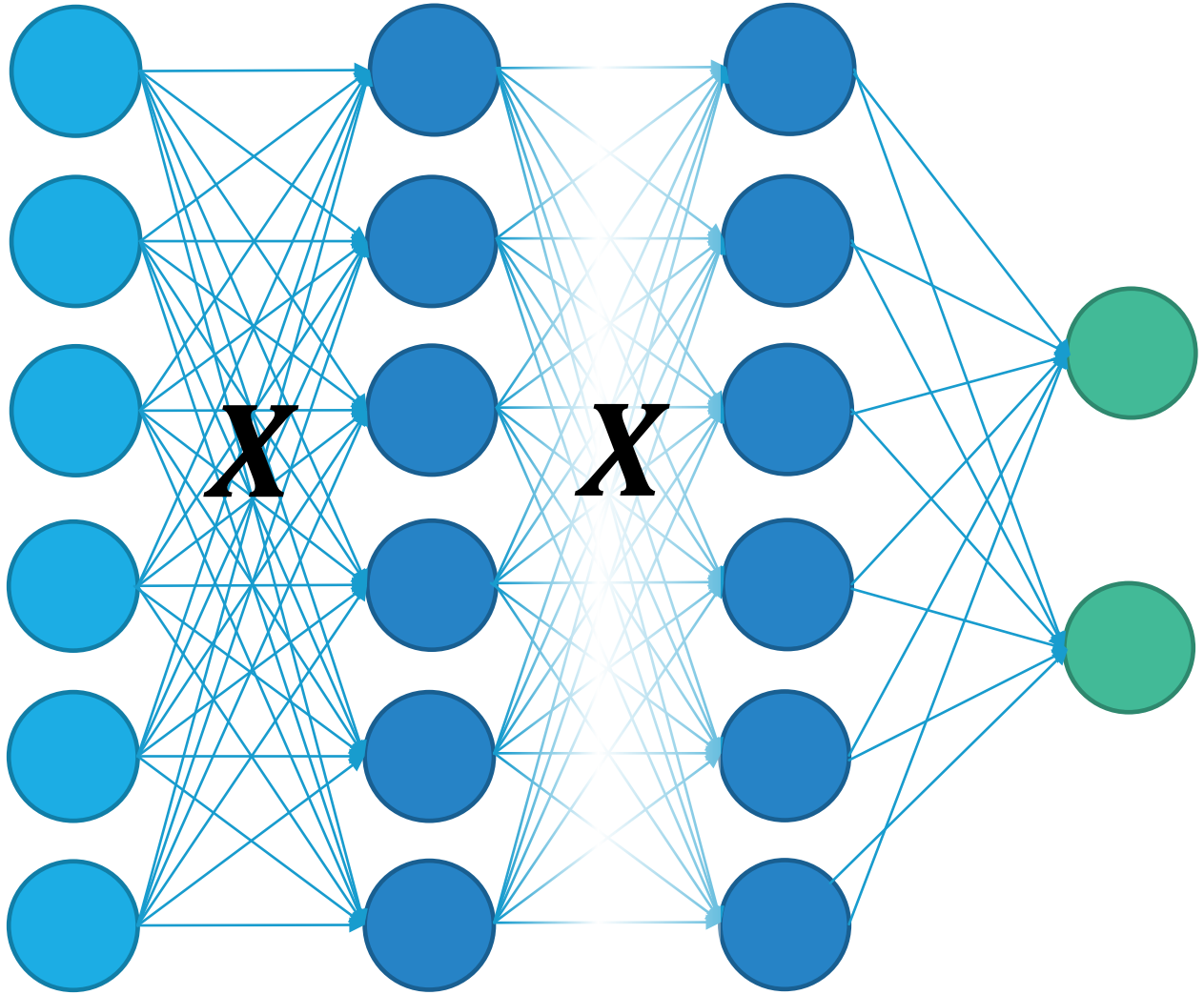
NEURAL NETWORK EXAMPLE

Input

Input layer

Multiple hidden layers

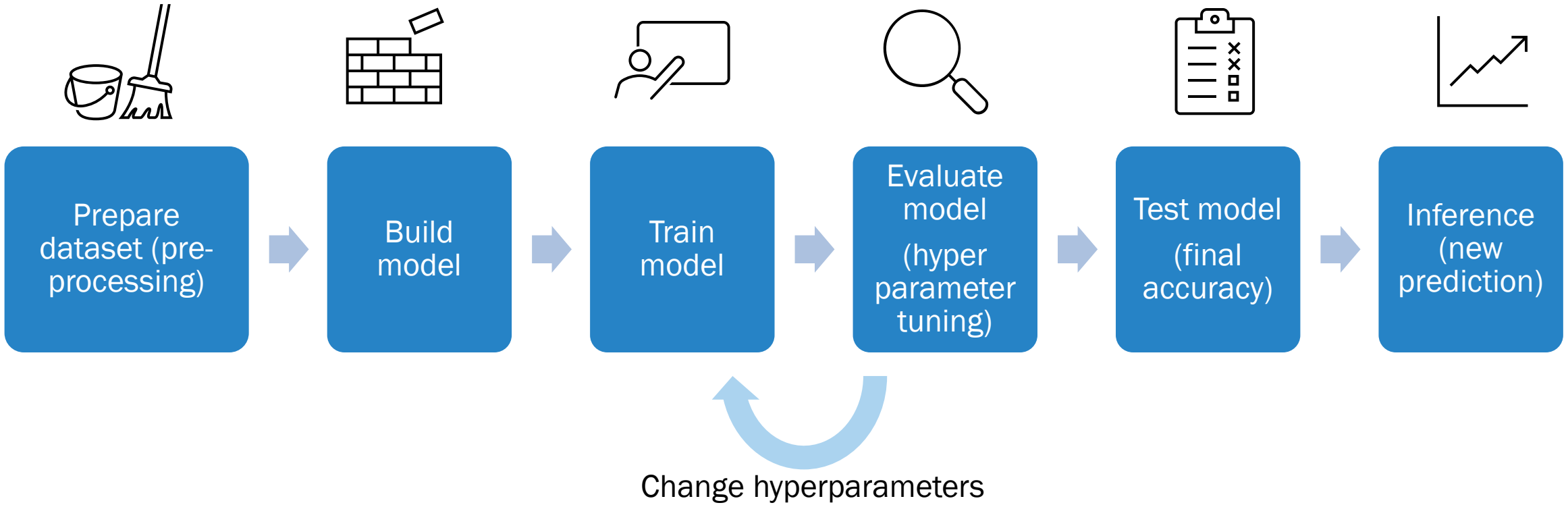
Output layer



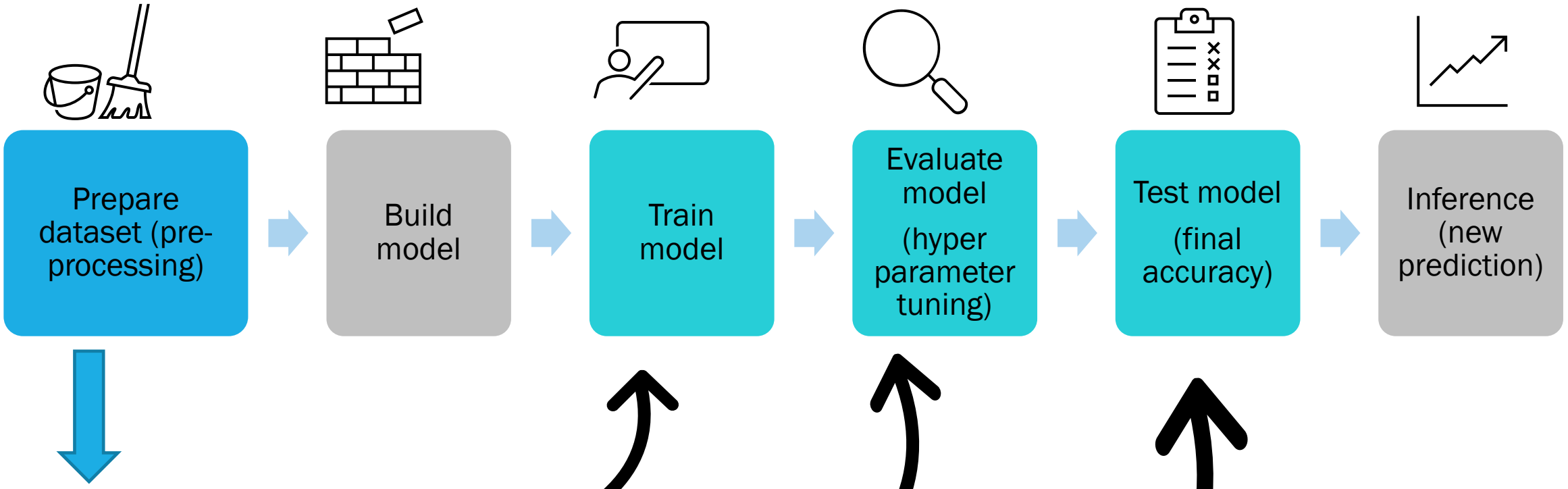
How do we initialise weights?

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

WORKFLOW


Prepare dataset (pre-processing)
Build model
Train model
Evaluate model (hyper parameter tuning)
Test model (final accuracy)
Inference (new prediction)
Change hyperparameters

WORKFLOW



- **Training set** —for training the model
- **Validation set** —for tuning the hyperparameters and evaluate the models
- **Test set** —to test the model and find the accuracy

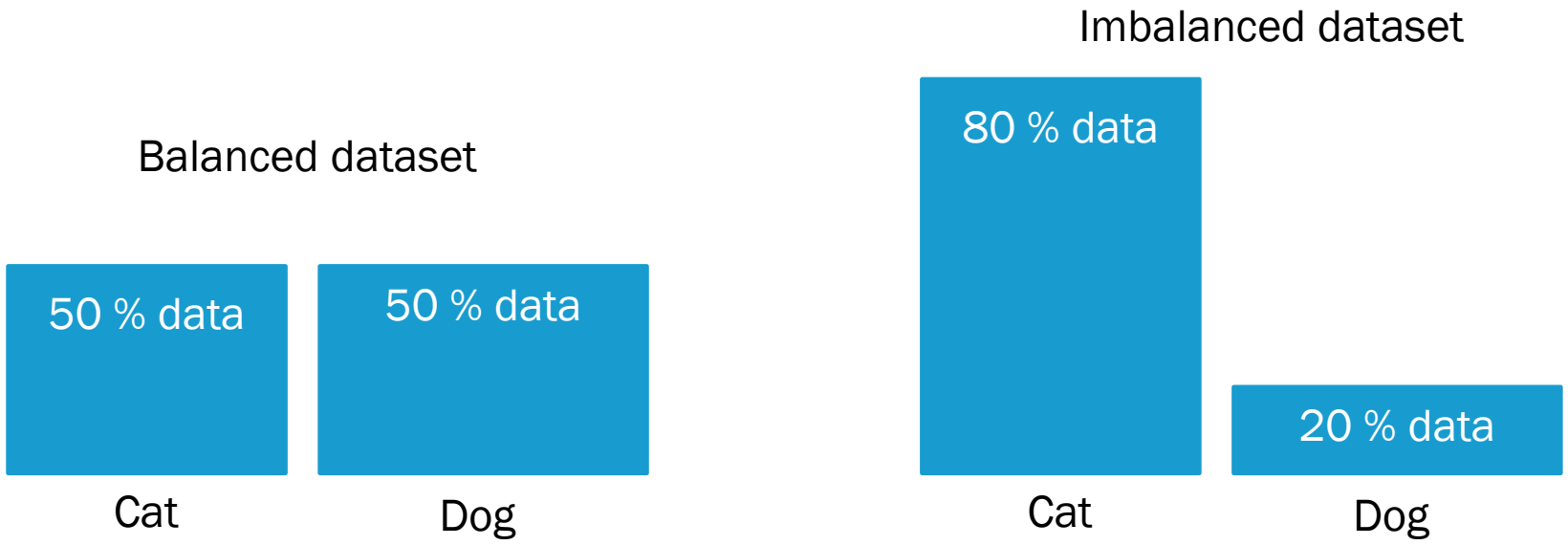
DATA

Prepare data (pre-processing)

- Data split

80%	10%	10%	(small datasets)
60%	20%	20%	(larger datasets)

- Know your data:



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]

REFERENCE: 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).

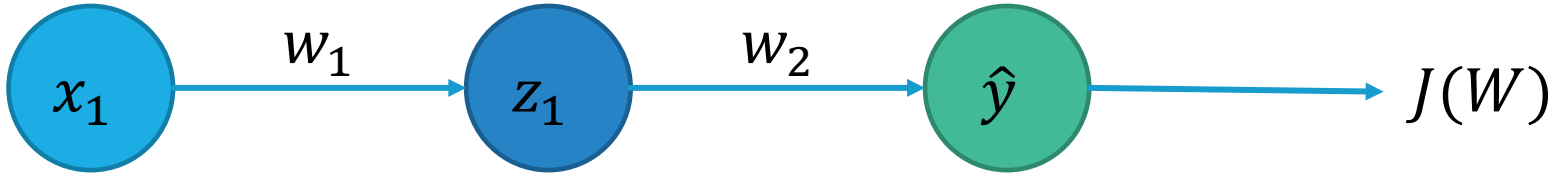
Binary classification	Multiclass classification	Regression
Loss: Cross Entropy	Loss: Cross Entropy	Loss: Mean Squared Error
Activation: Sigmoid	Activation: Softmax	

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

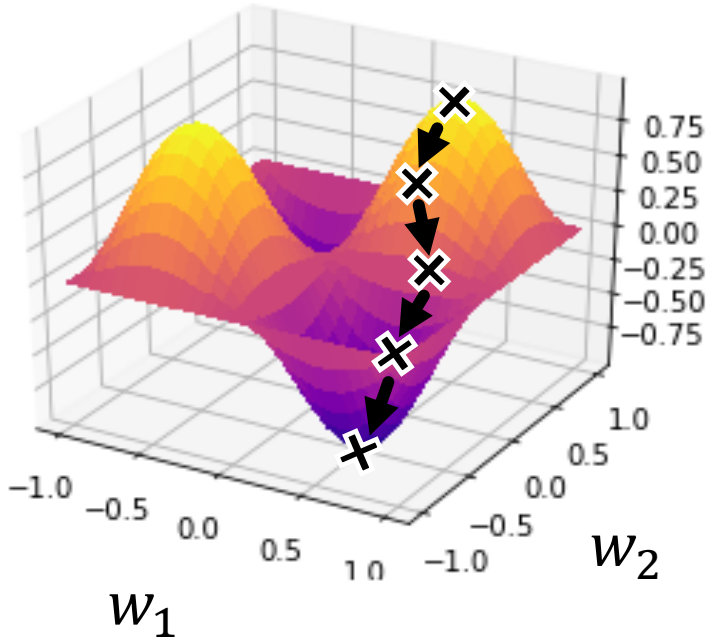
How much a change in w_2 will affect the total error (loss) $J(W)$?

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) $J(W)$?

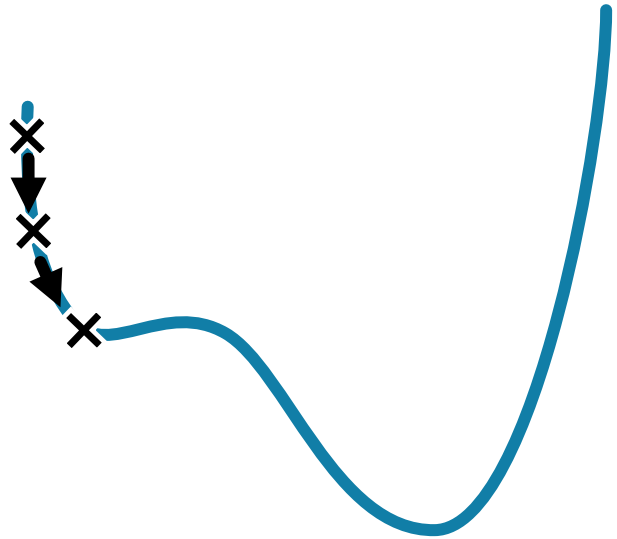
$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$



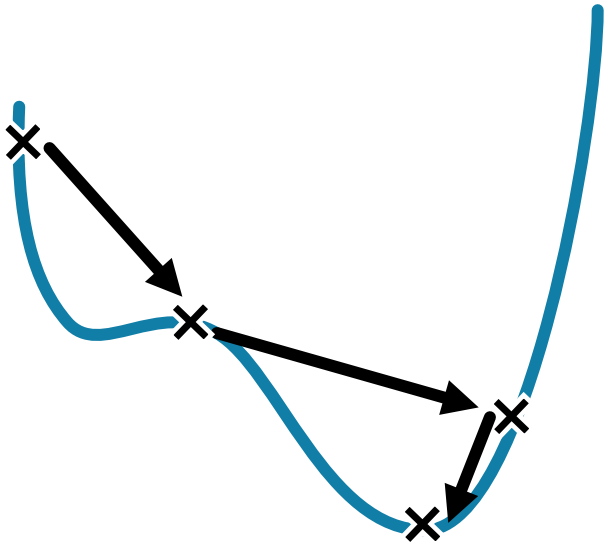
LEARNING RATE

Train model

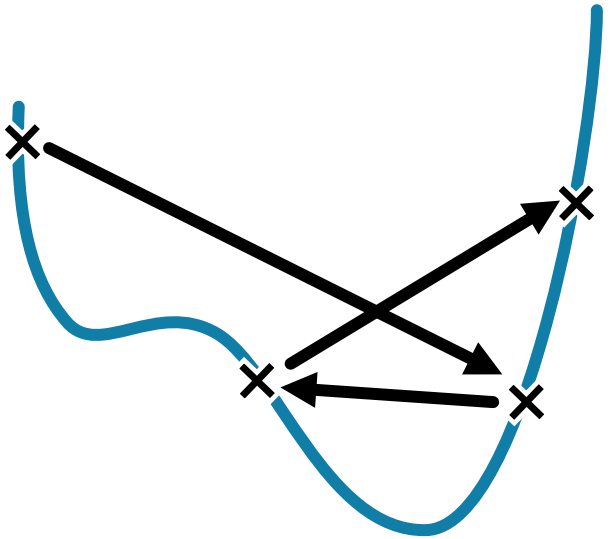
- How much step we take in the gradient?



Small learning rate



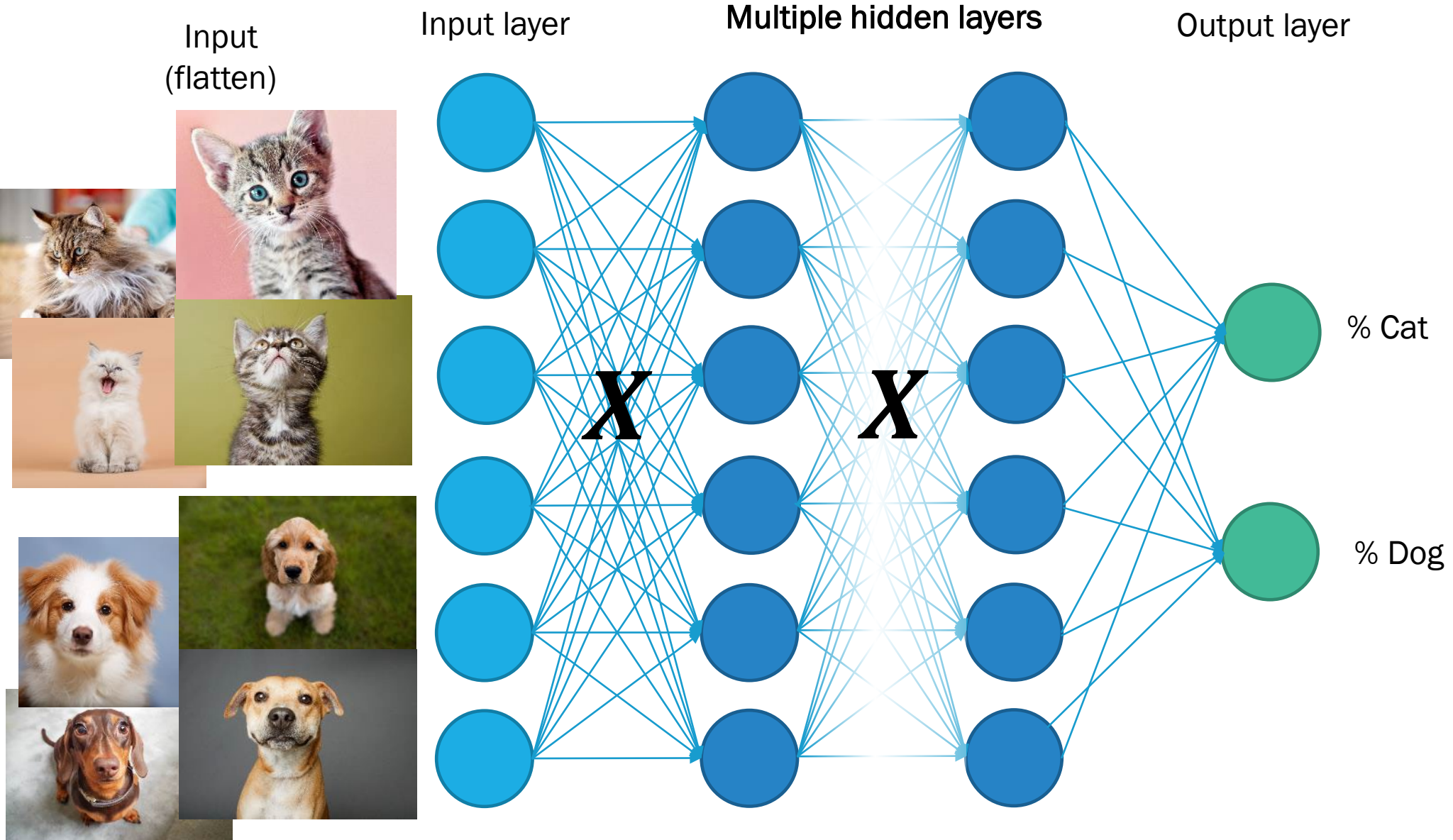
Good learning rate



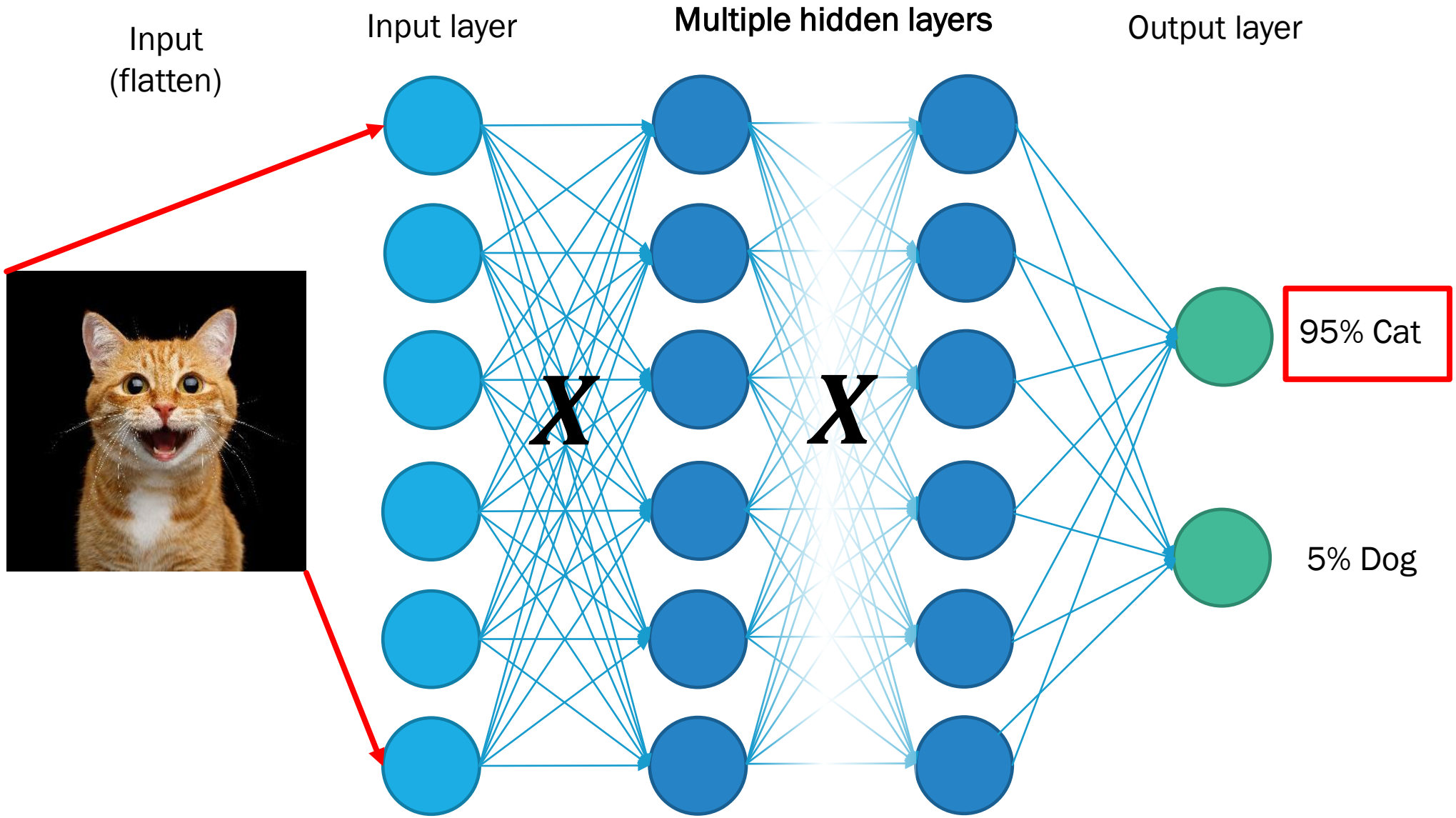
Large learning rate

FORWARD PROPAGATION AND BACKPROPAGATION

Train model



FORWARD PROPAGATION AND BACKPROPAGATION



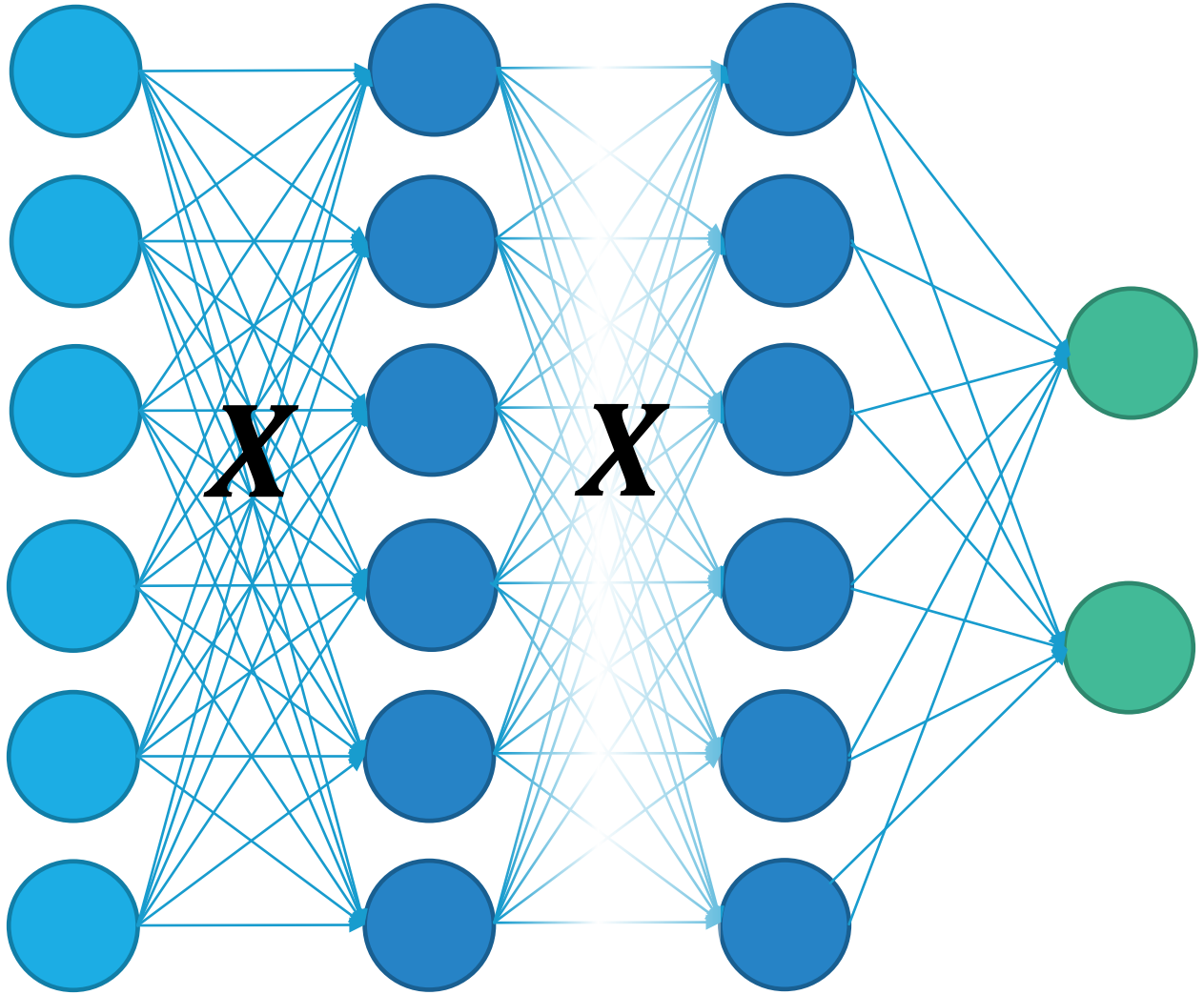
FORWARD PROPAGATION AND BACKPROPAGATION

Input
(flatten)

Input layer

Multiple hidden layers

Output layer



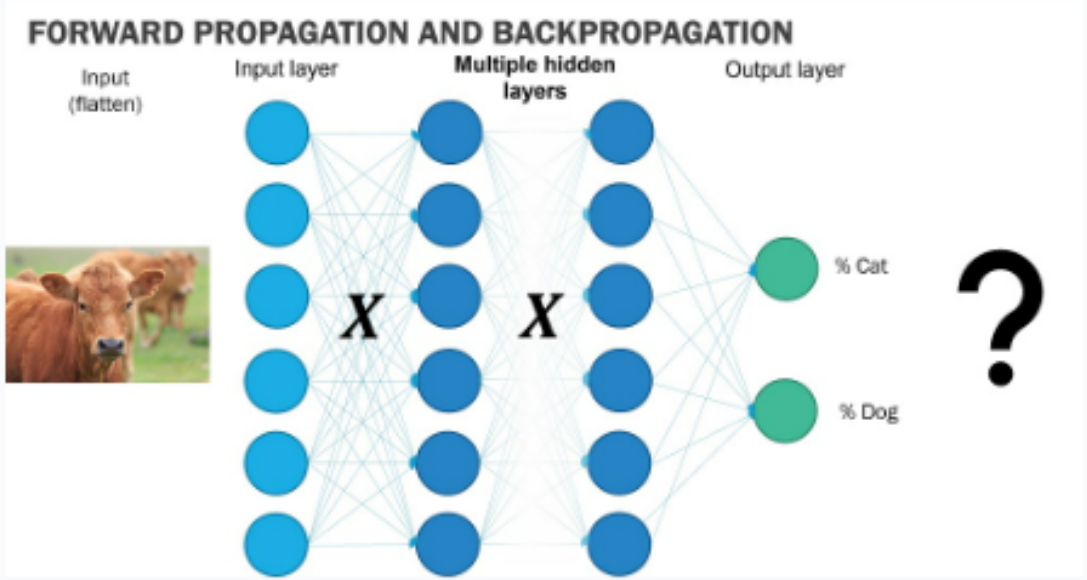
% Cat

% Dog

?

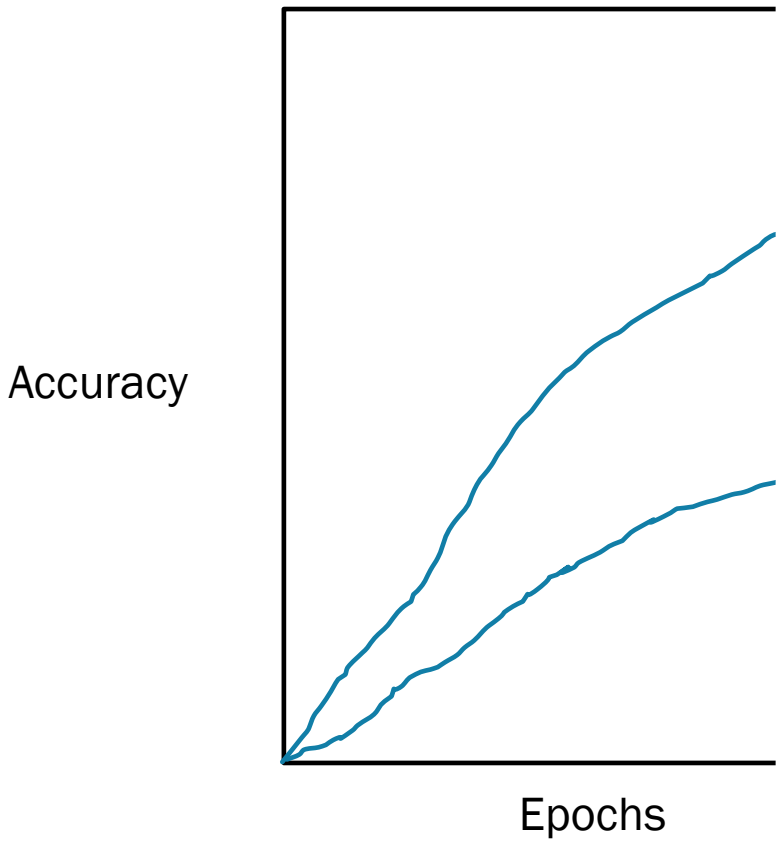
When poll is active, respond at **PollEv.com/esterbonmati**

What will the NN predict?



- Is a cow
- Is a cat
- Is a dog
- Will not make a prediction

TRAINING



Activities

When poll is active, respond at **PollEv.com/esterbonmati**

What would you do next?

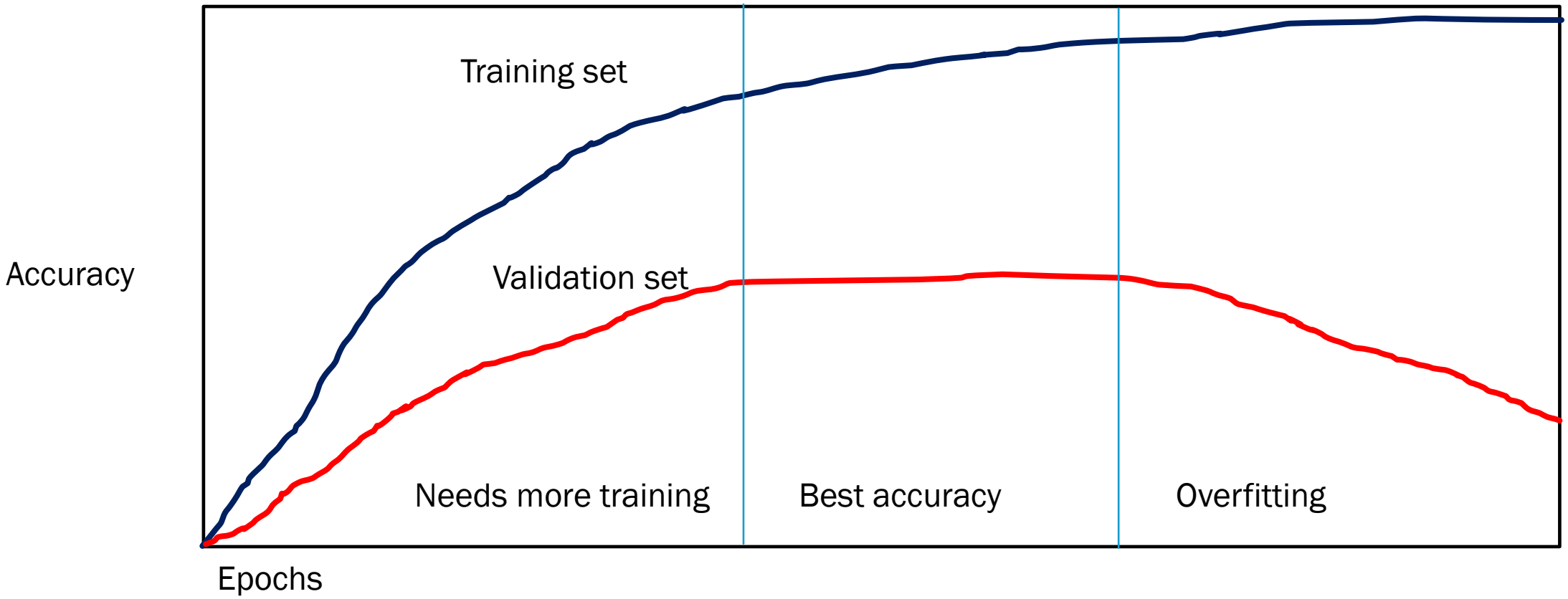
Decrease the learning rate

Increase the number of epochs

Powered by Poll Everywhere

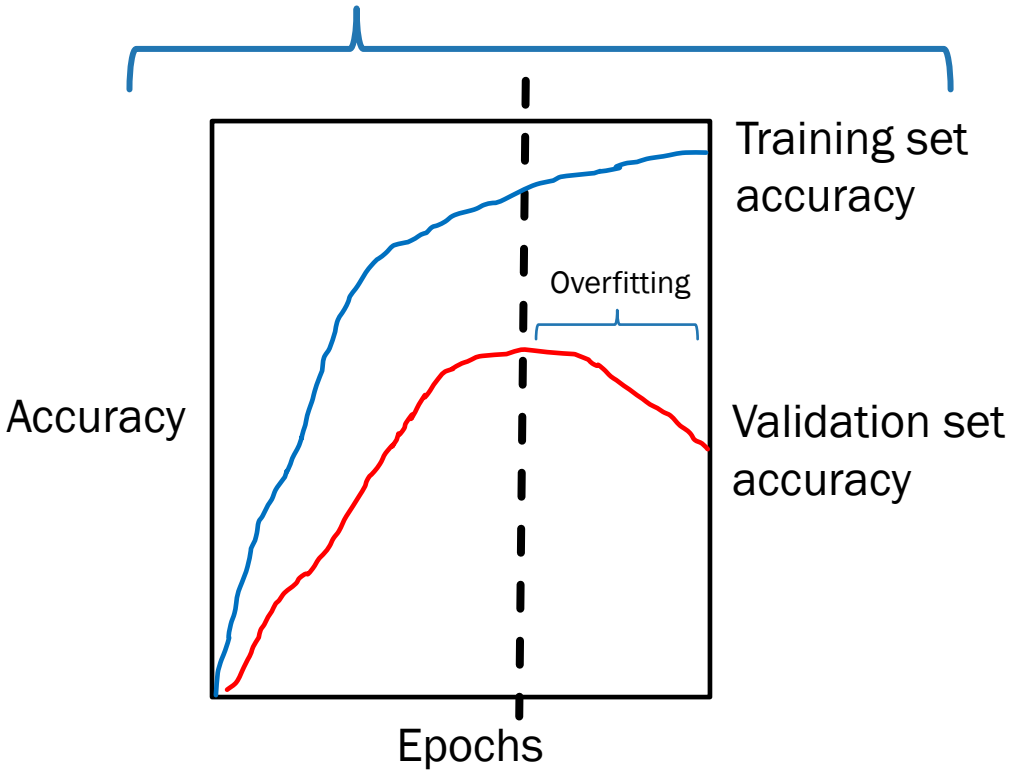
TRAINING

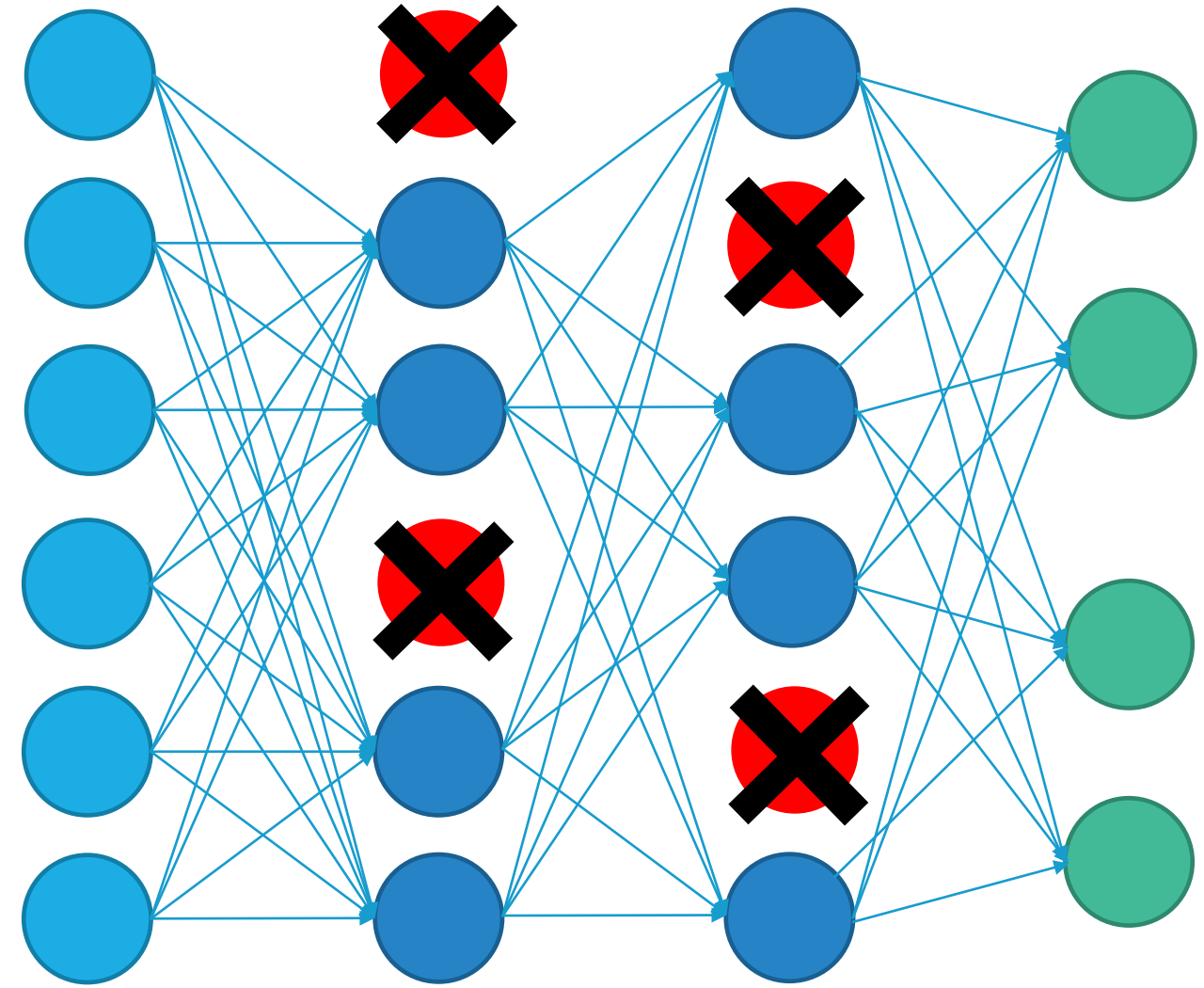
Train model



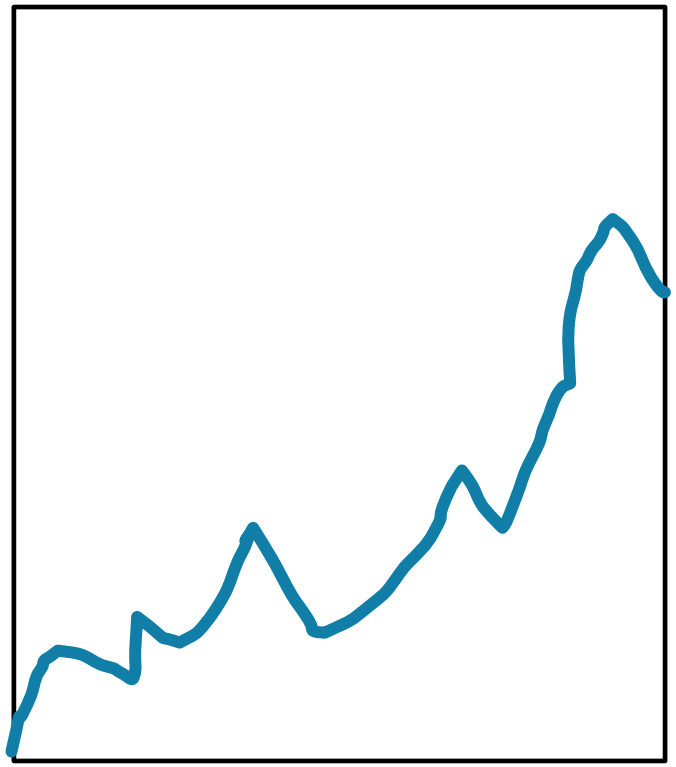
PROBLEMS: OVERFITTING

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





WHAT TO TRY NEXT?



Activities

When poll is active, respond at **PollEv.com/esterbonmati**

What would you do next?

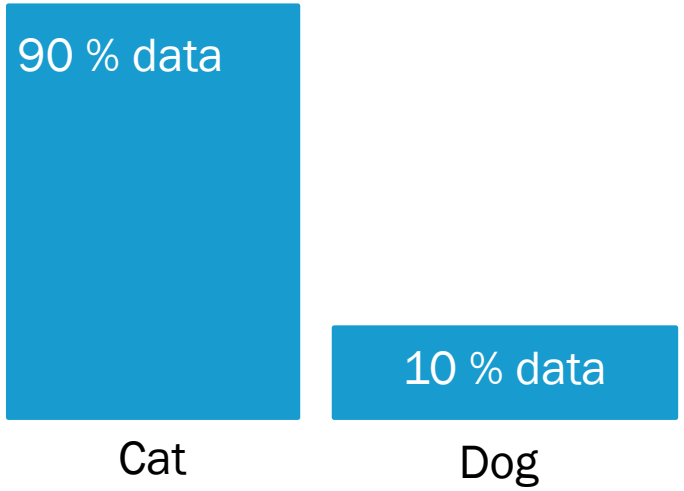
- Decrease the learning rate
- Increase the learning rate
- None of the above

Powered by Poll Everywhere

PROBLEMS WITH IMBALANCED DATASETS: ACCURACY

Prepare data (pre-processing)

Imbalanced dataset (100 samples)



If we predict everything as a cat....

What is for you a good accuracy?

$$accuracy = \frac{number\ correct\ samples}{total\ number\ of\ samples}$$

$$accuracy = \frac{90}{100} = \mathbf{0.9}\ (\mathbf{90\%}!!)$$

This model would have a 90% accuracy even if predicts everything as a cat.

Inputs and outputs

Overfitting

Weights

Forward propagation

Activation function

Gradient

Hyper parameters

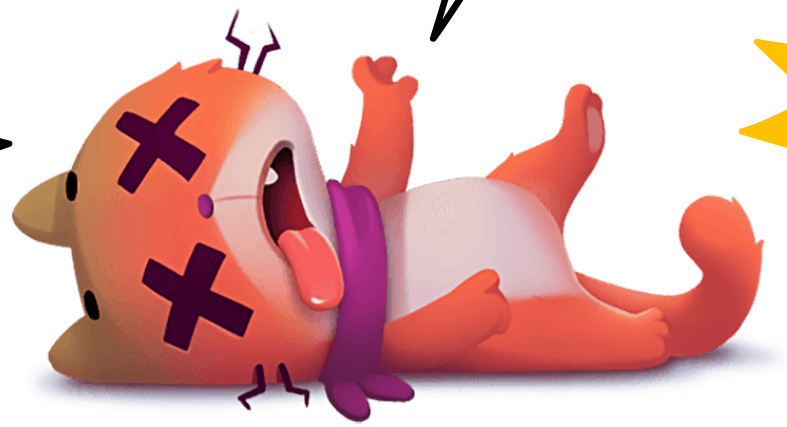
Hidden layer

Back propagation

Learning rate

Data split

Data

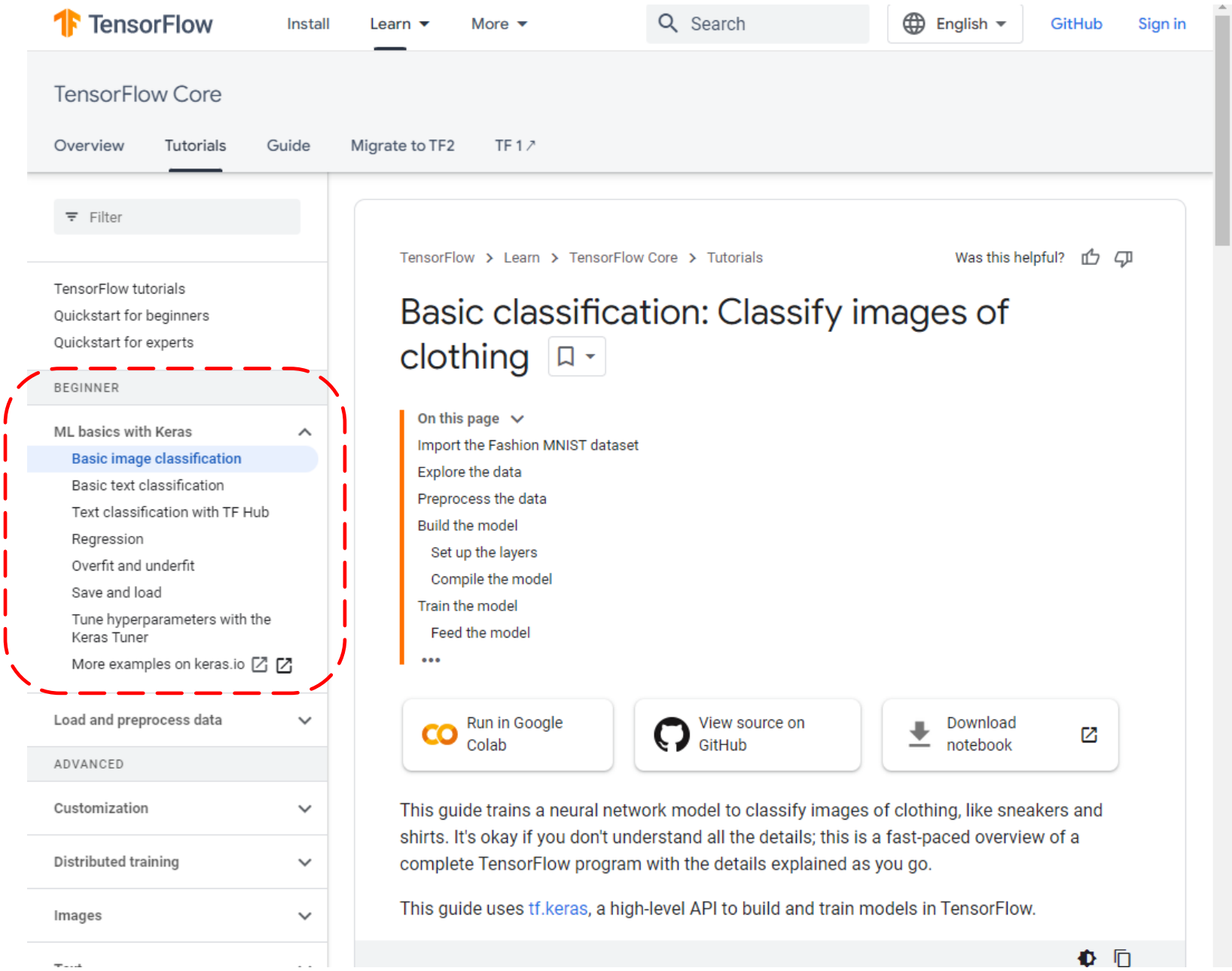


Epochs

Concepts to remember

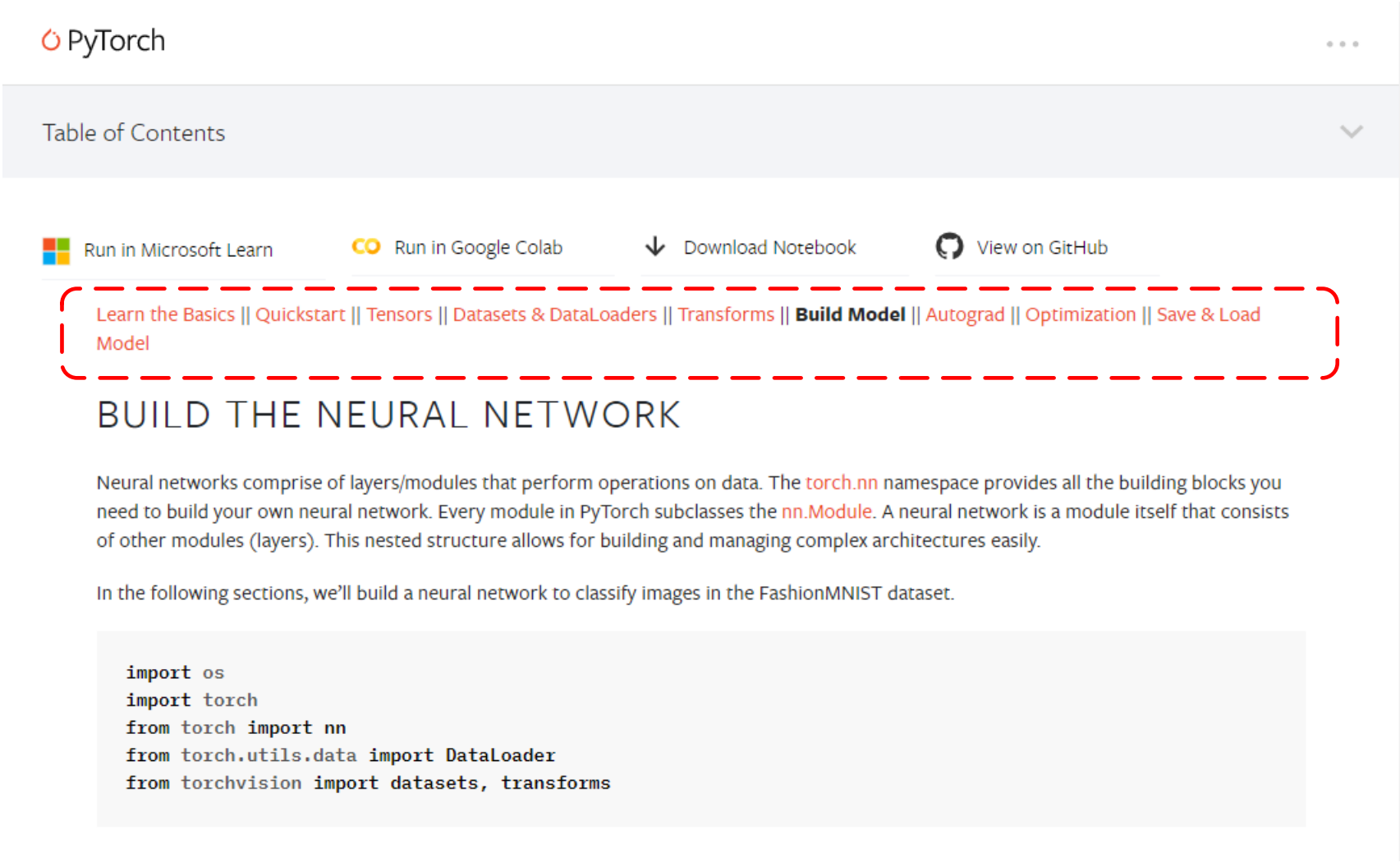
TENSORFLOW & KERAS

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



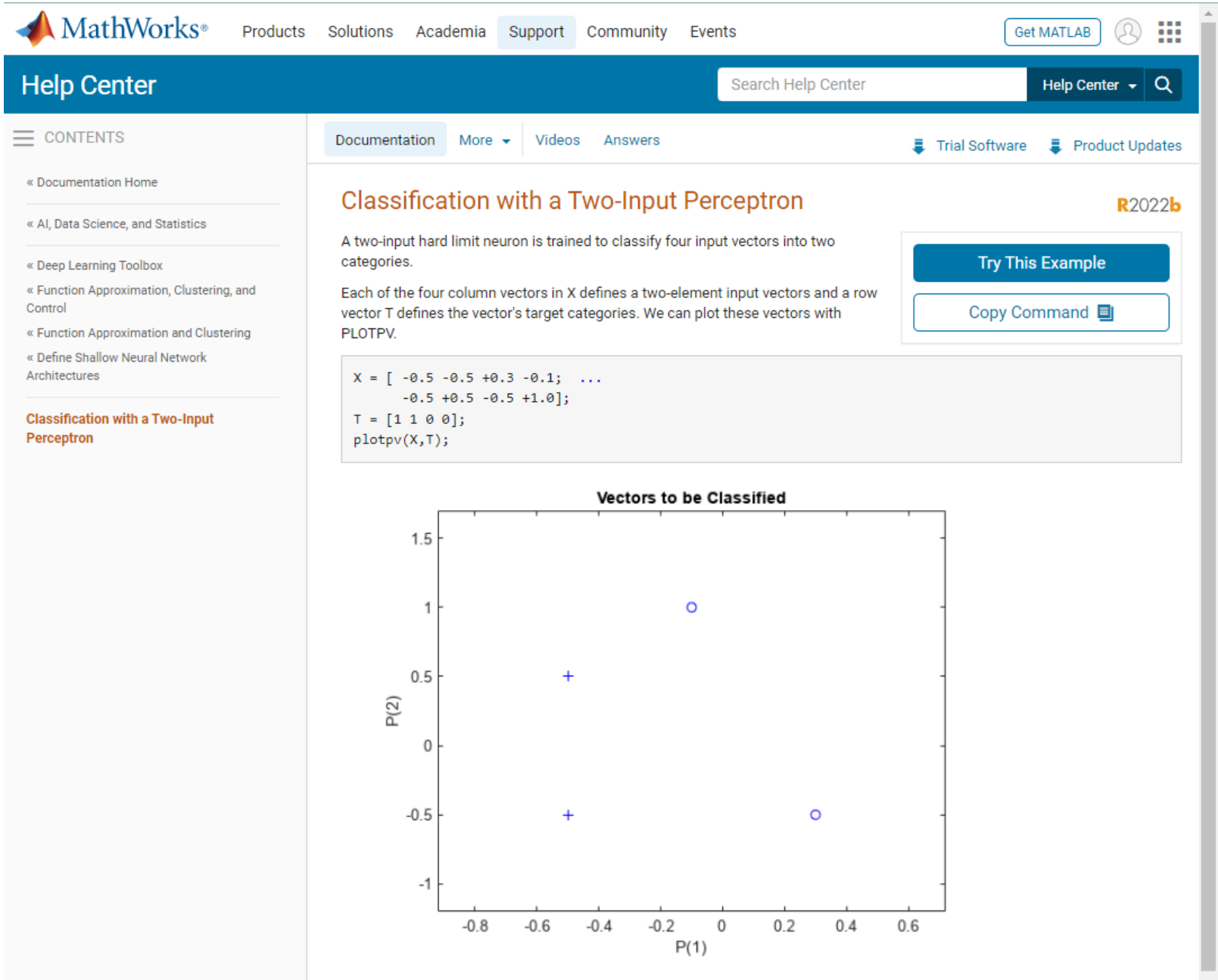
PYTORCH

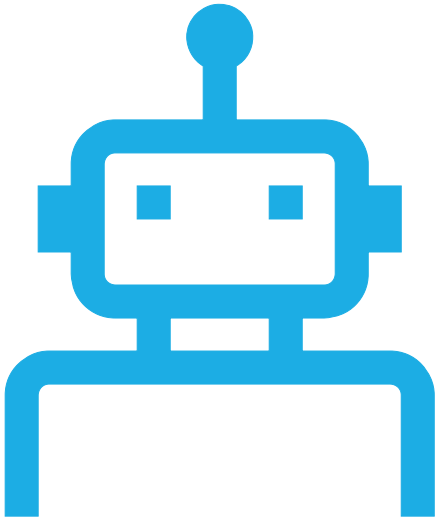
- https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html



ALSO IN MATLAB

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





TUTORIAL: SUPPORTING MATERIAL

NEURAL NETWORKS: A PRACTICAL EXAMPLE WITH TENSORFLOW

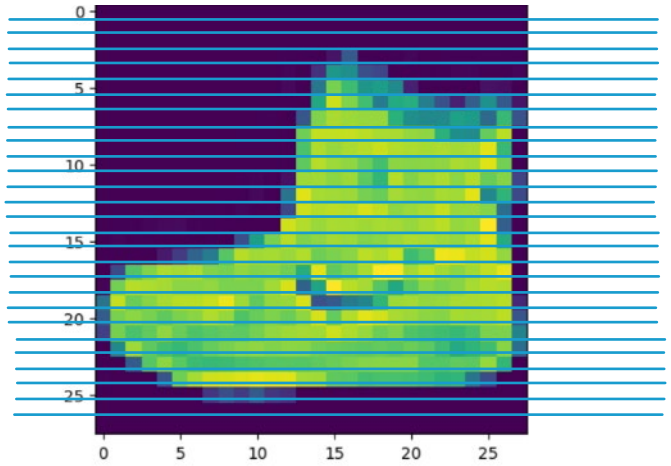
- Example with TensorFlow
- We will work with this example in the Tutorial

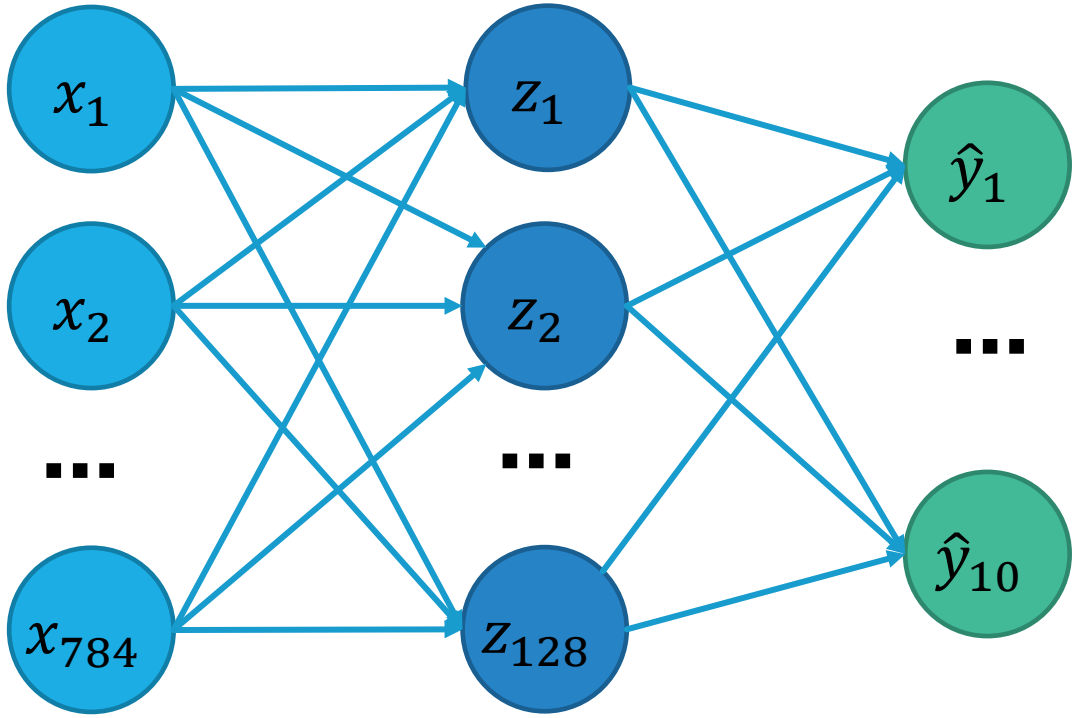

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

fashion_mnist = tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

MODEL CREATION






```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10) ])
```

Input = 28*28 = 784

Output

Hidden layer with 128 units and a ReLu activation function

Total trainable parameters: (784*128) + 128 + (128*10) + 10 = **101,770**

SET OPTIMISER AND TRAINING

Optimiser

Learning rate

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Loss

Training set

Validation set

```
epochs = 10
history = model.fit(train_images, train_labels, validation_data=[test_images, test_labels],
                    epochs=epochs)
```

Number of epochs (iterations)

TEST & INFERENCE (PREDICTION)

Softmax to normalise the logits (output) into probabilities

```
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

Array of predicted values for each test image

Make prediction

Images to predict

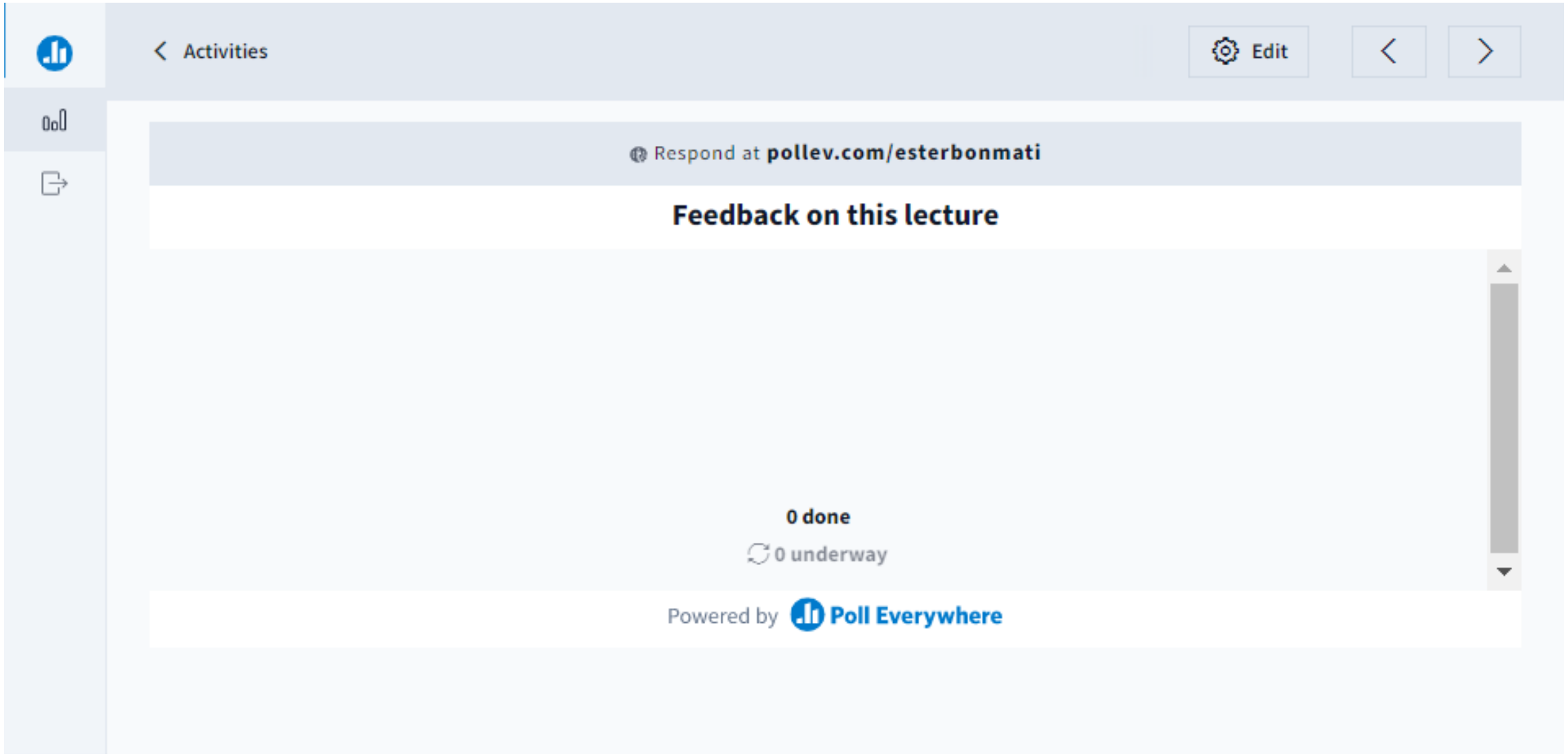
When poll is active, respond at **PollEv.com/esterbonmati**

Q&A

Top

No responses received yet. They will appear here...

FEEDBACK ON THIS LECTURE



LEARNING OUTCOMES OF THIS SESSION

Perceptron

- Understand the fundamental concept
- Understand inputs and outputs
- Understand weights
- Forward propagation
- Understand activation functions
- Be able to calculate the output of a perceptron

Neural Networks

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

ACKNOWLEDGEMENTS & NOTES

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)

Notes:

- We have some ‘guest’ students attending this lecture.
- Anne-Gaelle Colom (Assistant Head of School) will observe the second hour of the tutorial.