



UCL ARTIFICIAL
INTELLIGENCE SOCIETY

TUTORIAL #5

Session 5

Ensemble Methods and Neural Networks

While we wait, sign up at doxaai.com for this week's challenge - more at the end!

LECTURE OVERVIEW

01 |

Ensemble models

How can we make our simple models more useful?

02 |

Gradient descent

Recap of this incredibly important concept

03 |

Neural Networks

The most famous class of Machine Learning models

04 |

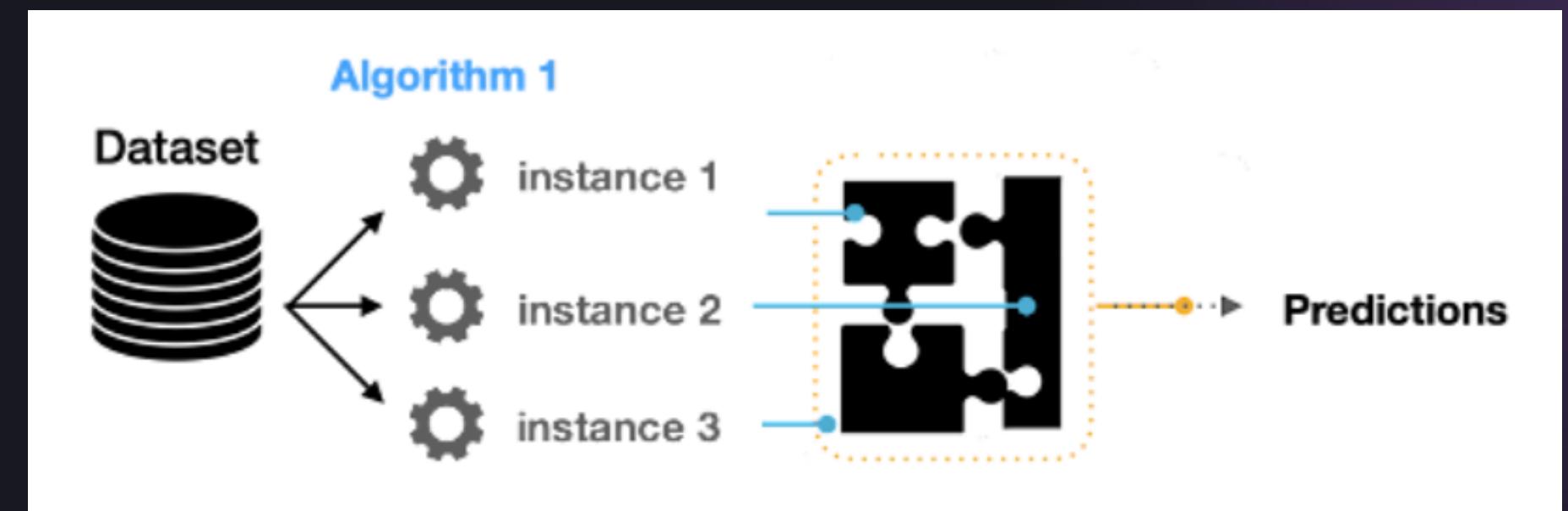
Competition

Try out everything you learned so far!

ENSEMBLE MODELS

What are Ensemble Models?

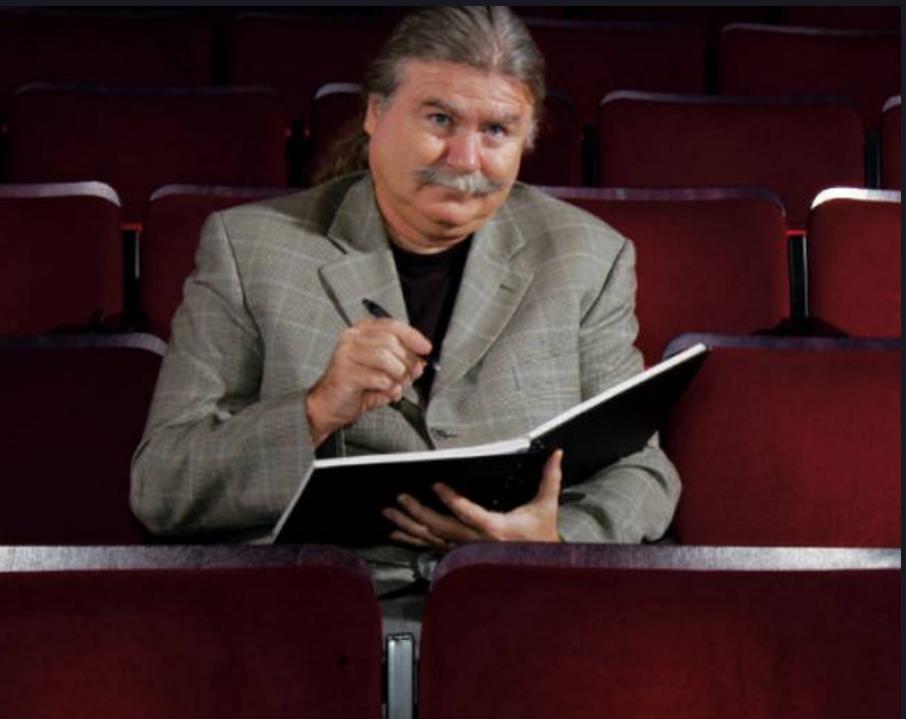
- They are a machine learning approach to **combine multiple other models** in the prediction process
- Singular model in EM is usually a weaker predictor than a standard supervised model



Theory

ENSEMBLE MODELS

What are Ensemble Models?



ENSEMBLE MODELS

Different approaches for Ensemble Models

Simple Ensemble Techniques

- Max Voting
- Averaging
- Weighted Averaging

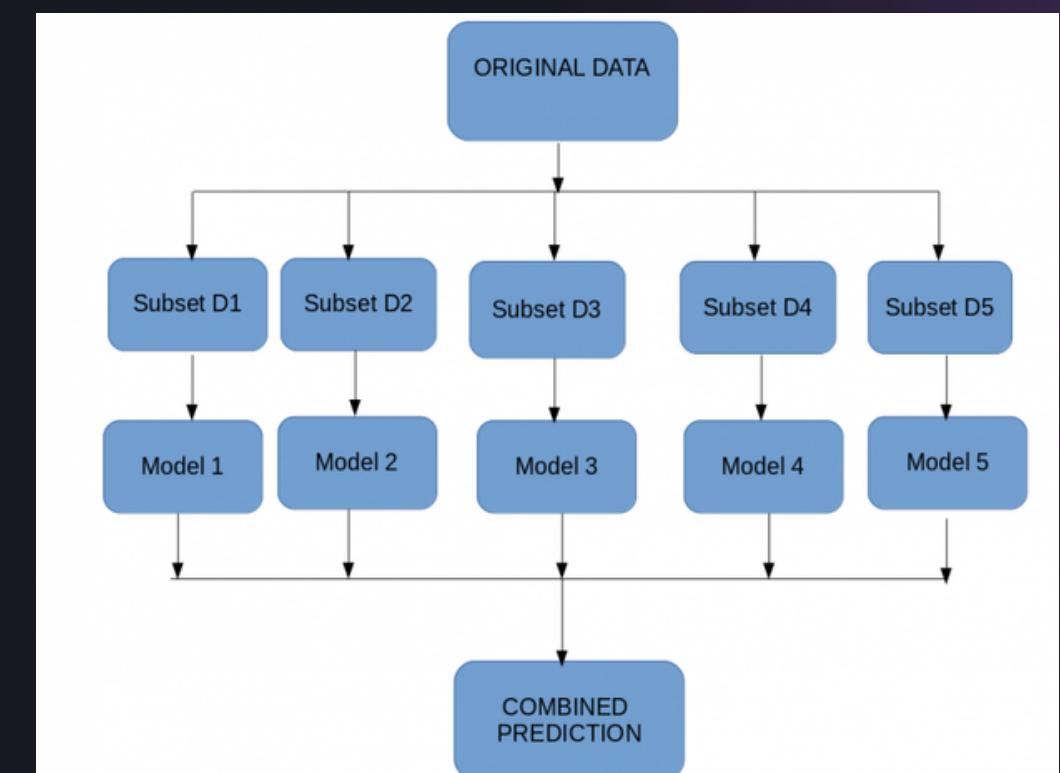
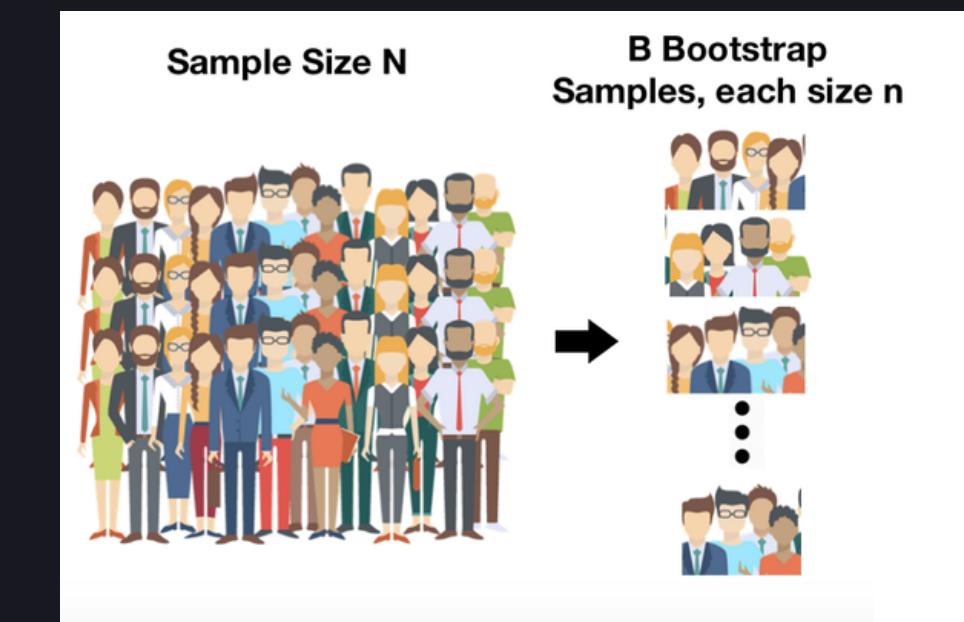
Advanced Ensemble Techniques

- Stacking
- Blending
- Bagging
- Boosting

ENSEMBLE MODELS

Bagging

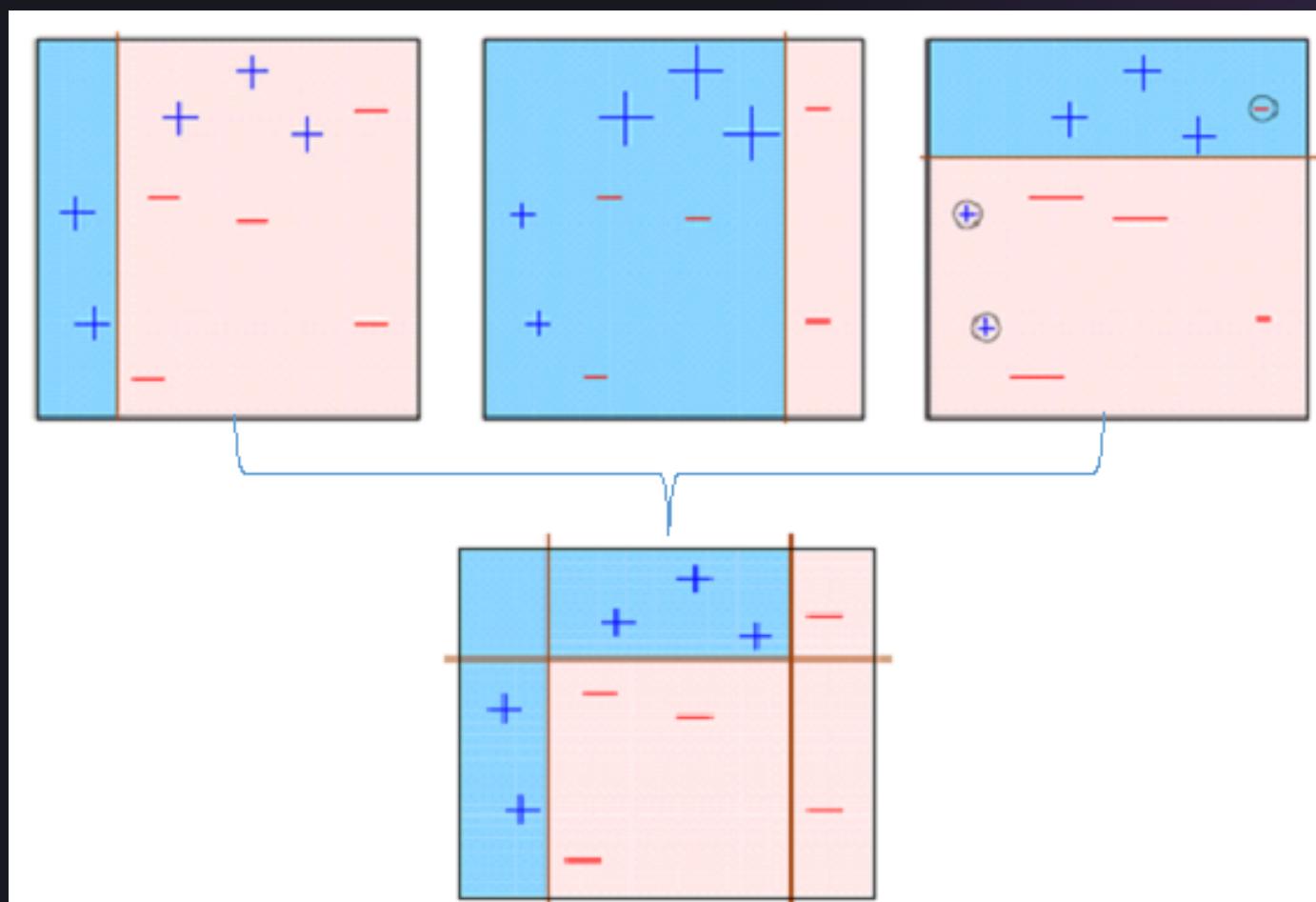
- Combining results of multiple models (e.g. decision trees)
- Uses the technique of **Bootstrapping**
 - Taking subsets, which overlap
 - Models are trained on these subsets
 - Predictions from these models are then combined



ENSEMBLE MODELS

Boosting

- Each model tries to **correct the errors of previous models**
- Initially, a **single model is trained** on a subset of the dataset
- It evaluates the whole dataset, and a **new subset for training a model is chosen** to counter the errors
- The final model is weighted mean of all the models



ENSEMBLE MODELS

Random Forest

- Consists of numerous decision trees trained on bootstrapped data
- Is the primary example of bagging approach

```
from sklearn.ensemble import RandomForestRegressor  
  
reg = RandomForestRegressor()  
reg.fit(X_train, y_train)  
reg.score(X_test, y_test)
```

ENSEMBLE MODELS

XGBoost

- XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm
- Gradient boosting is a version of boosting using Gradient Descent

```
from sklearn.ensemble import GradientBoostingRegressor  
  
xgboost = GradientBoostingRegressor(n_estimators=100)  
xgboost.fit(X_train, y_train)  
xgboost.score(X_test, y_test)
```

GRADIENT DESCENT

What is Gradient Descent

- Gradient Descent is an **optimization algorithm** used for **minimizing the cost function/loss function** in various machine learning algorithms
- It is used for updating the parameters of the learning model.

GRADIENT DESCENT

Cost functions/Loss function

- Loss function aims to **capture the difference between the actual and predicted values** for a single record whereas cost functions aggregate the difference for the entire training dataset.

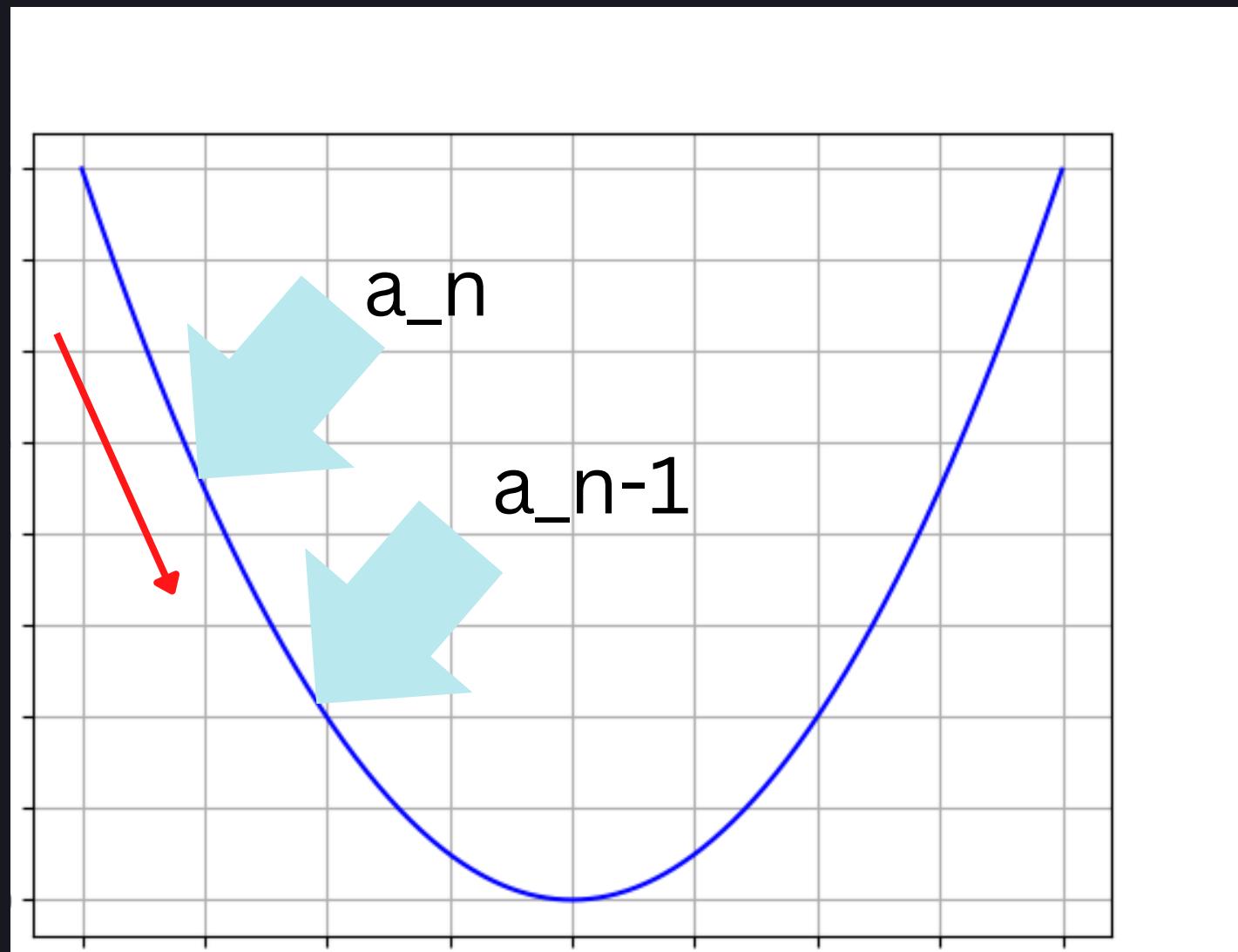
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

$$\text{MAE} = \underbrace{\frac{1}{n} \sum_{i=1}^n}_{\text{test set}} |y_i - \hat{y}_i|$$

predicted value actual value

Theory

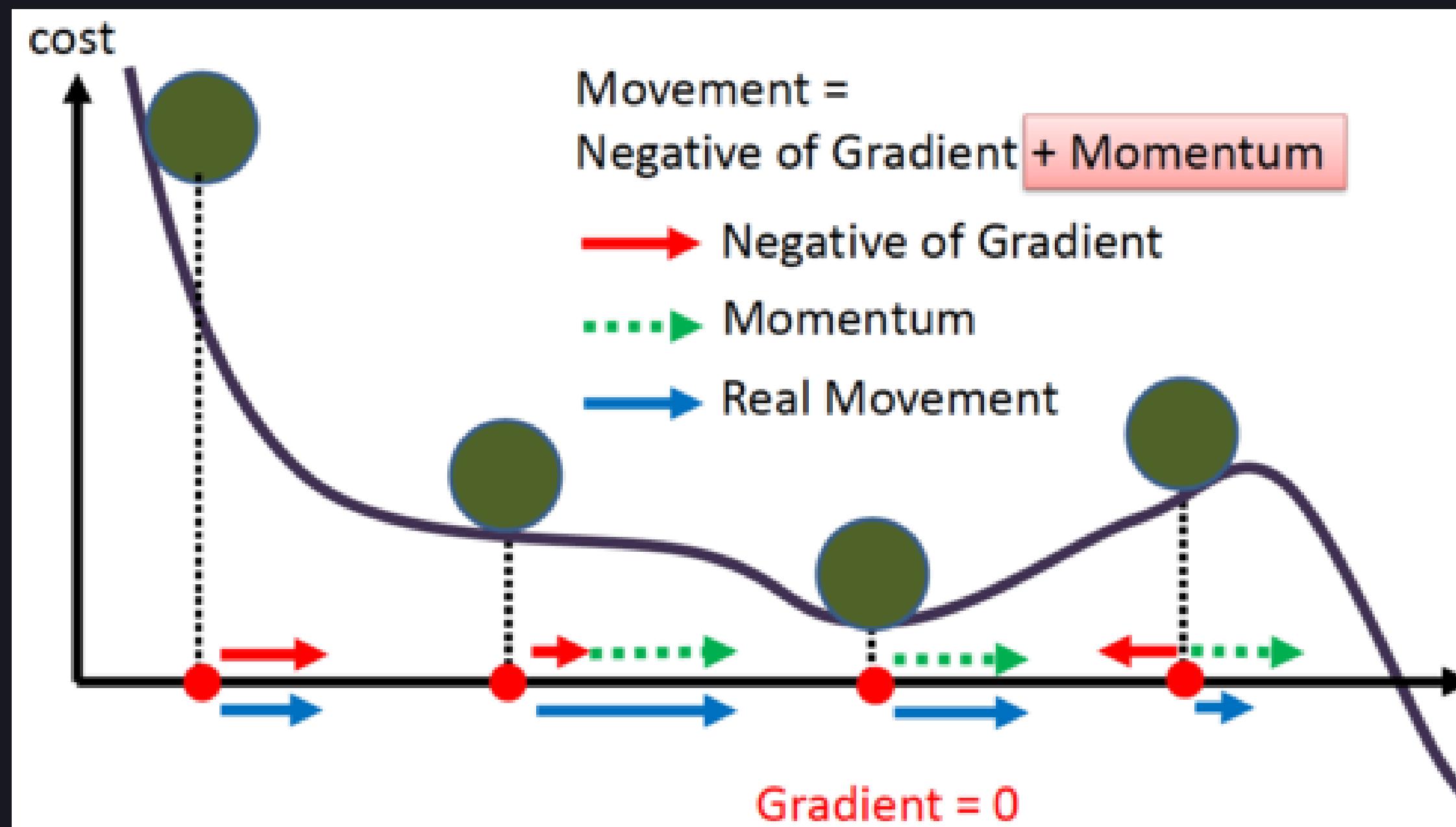
GRADIENT DESCENT



$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

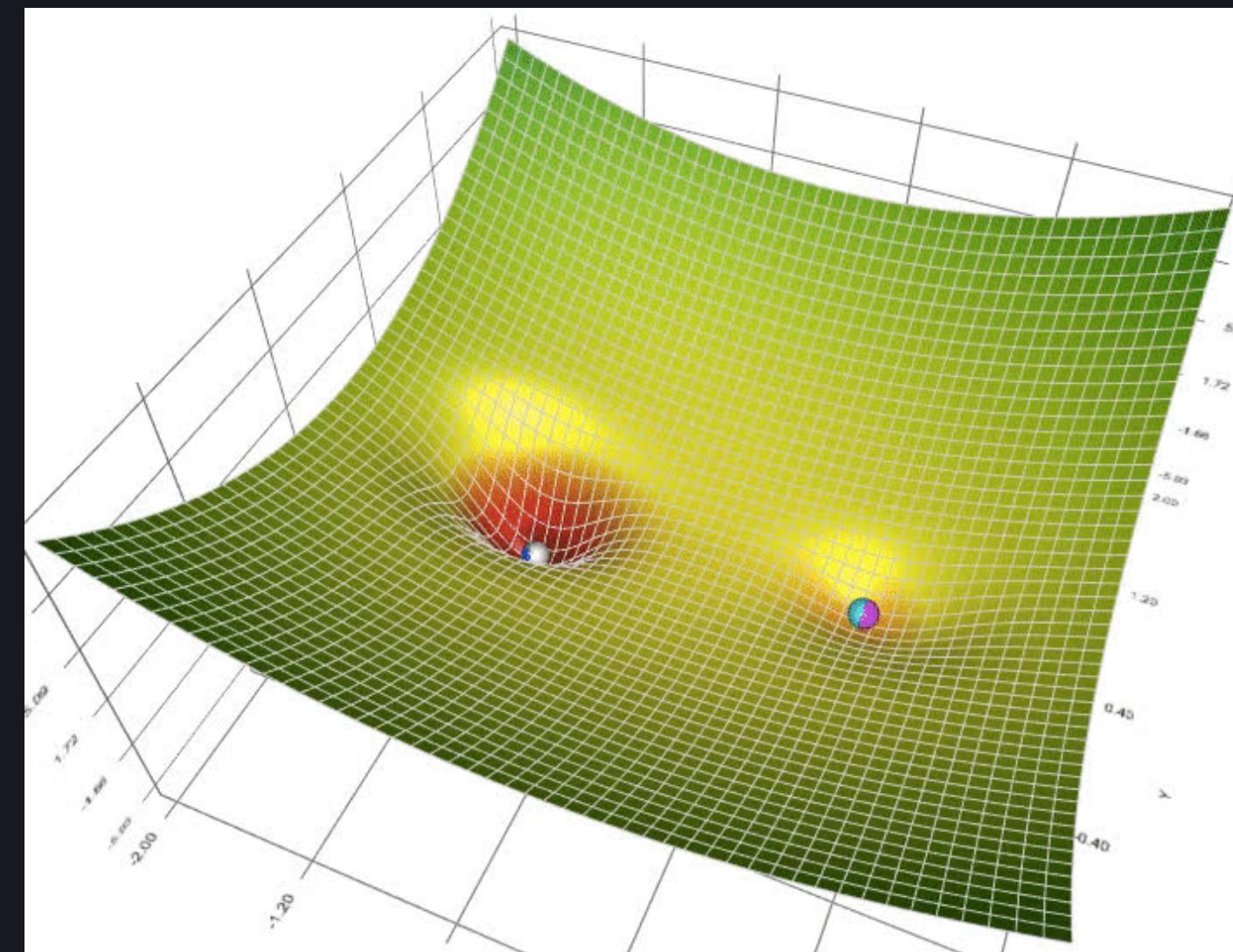
Theory

GRADIENT DESCENT



Theory

GRADIENT DESCENT



NEURAL NETWORKS

Introduction

- A method of computation based on the interaction of many connected processing elements called neurons.
- A powerful technique for solving many real-world problems.
- The ability to learn from experience in order to improve their performance.
- Ability to deal with incomplete information

Theory

NEURAL NETWORKS

Basics of Neural Networks

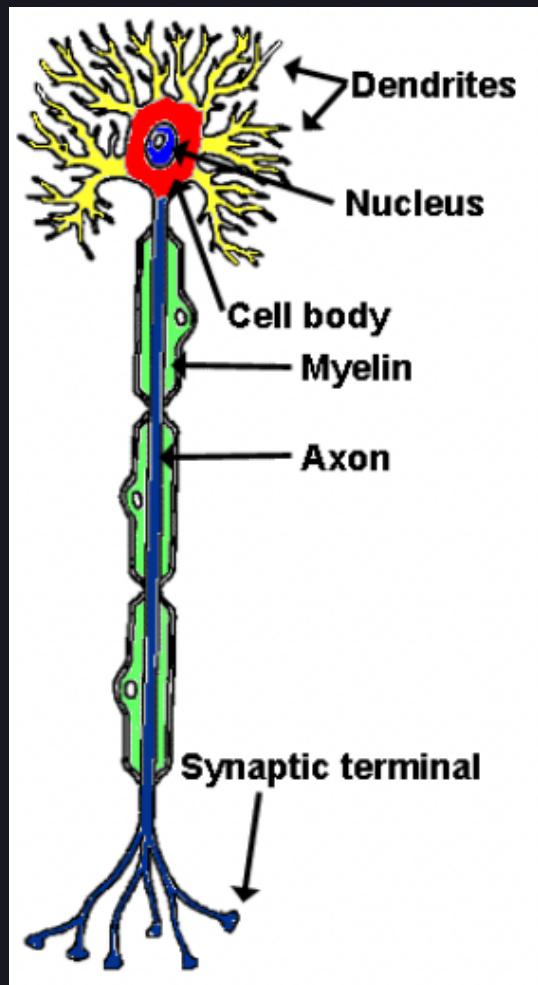
- Biological approach to AI
- Developed in 1943
- Composed of one or more layers of neurons
- Several types, we'll focus on feed-forward and feedback networks

Theory

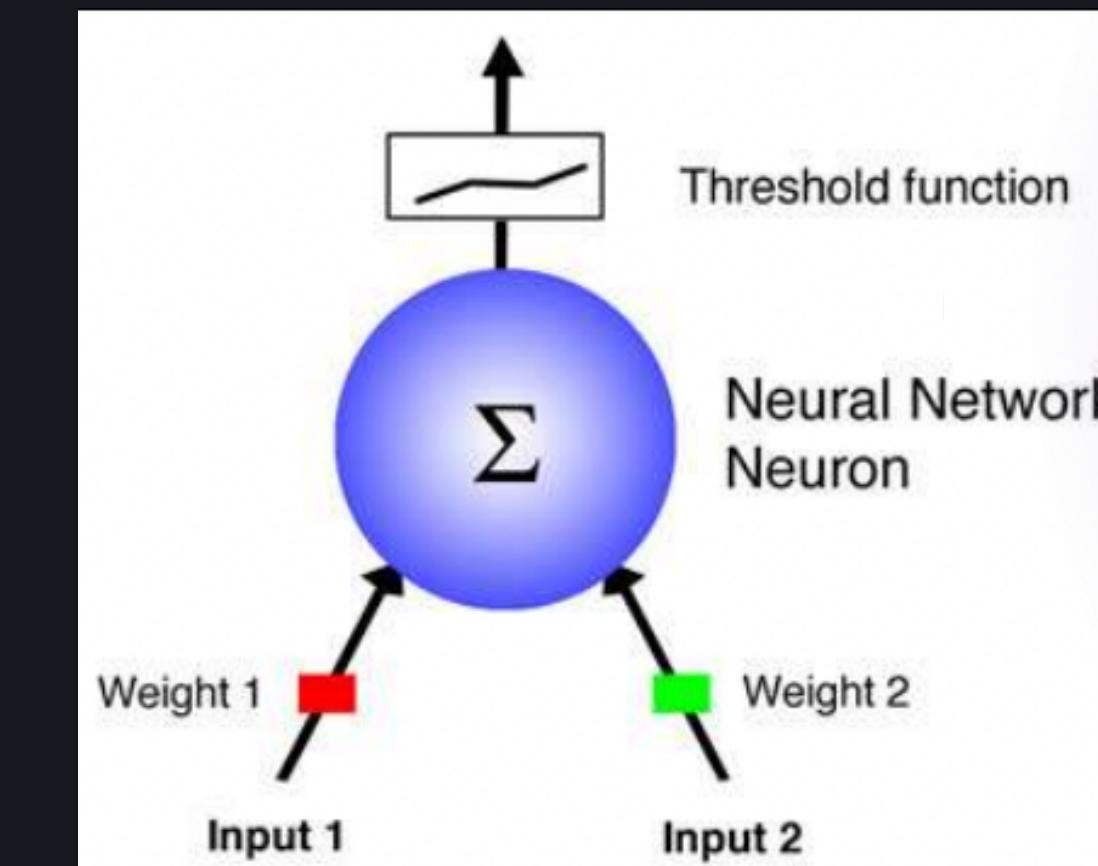
NEURAL NETWORKS

Neurons

Biological



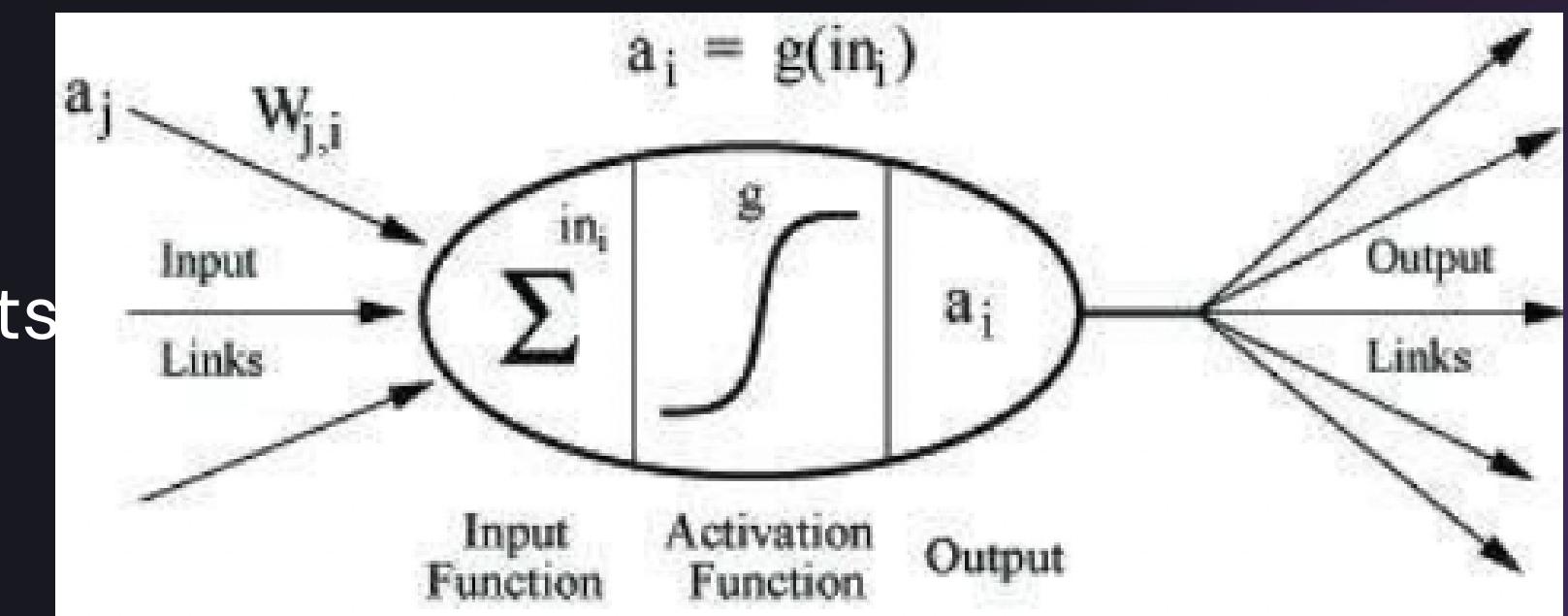
Artificial



NEURAL NETWORKS

Artificial Neurons

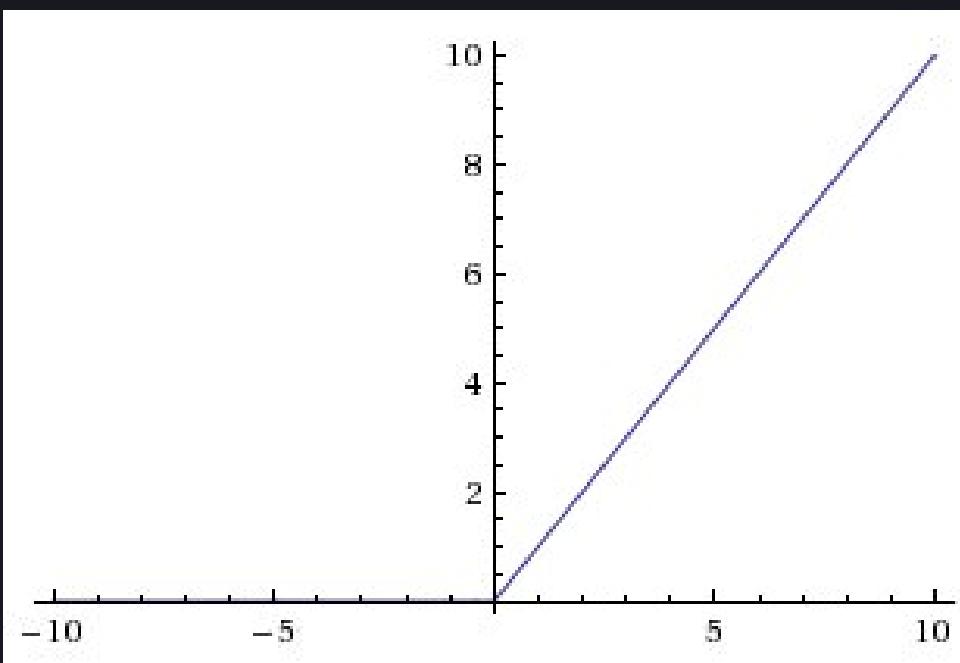
- Receive n-inputs
- Multiply each input by a weight
- Apply an **activation function** to the sum of results
- Output a result



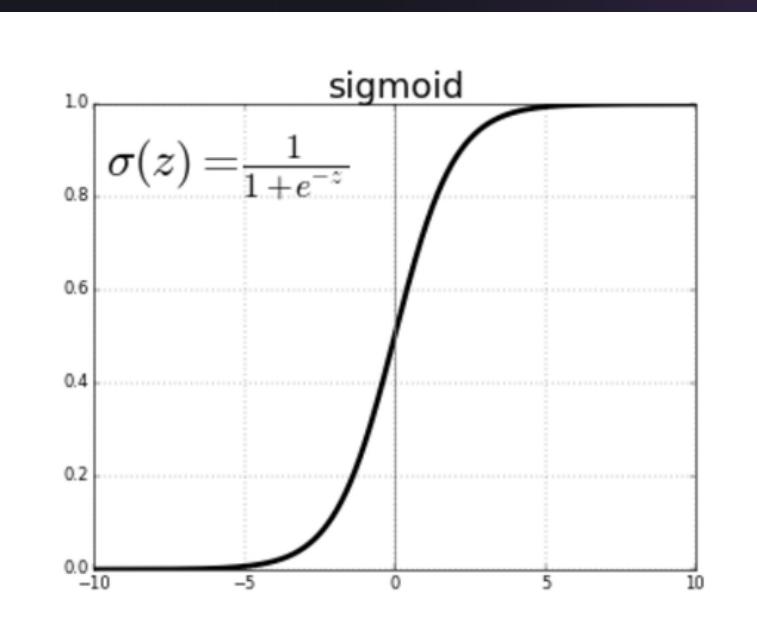
NEURAL NETWORKS

Activation Functions

- Transform the output of neurons
- Control 'active' or 'inactive' a unit is
- Allow non-linear relationships to be modelled



- **Rectified linear unit:** $\text{ReLU} = \max(0, x)$
- **Sigmoid function:** $\sigma = 1 / (1 + \exp(-x))$

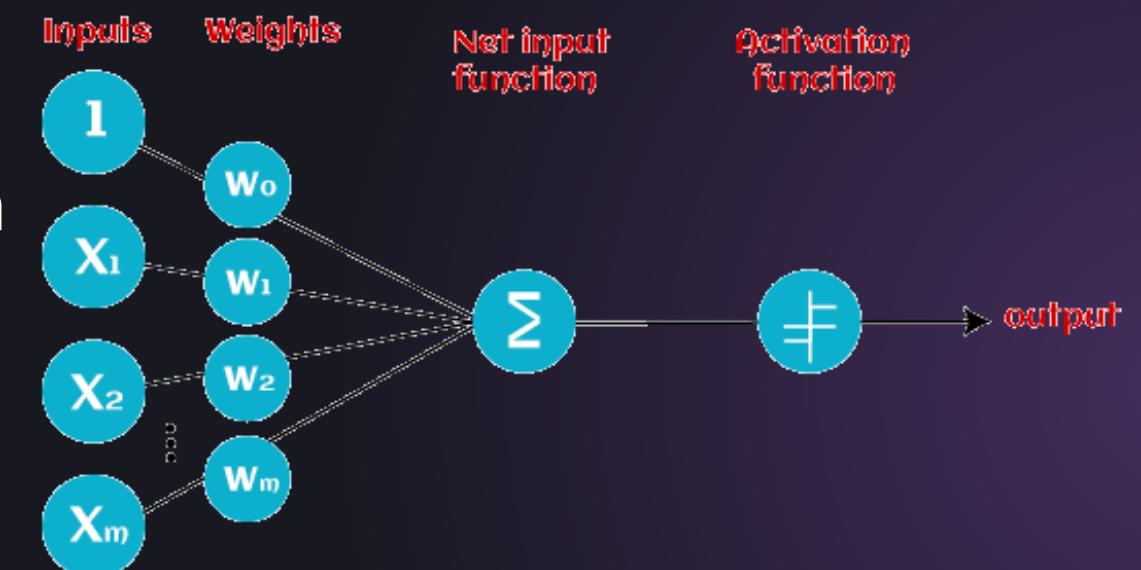


NEURAL NETWORKS

Perceptron

Frank Rosenblatt introduced the perceptron model, which contains three main components:

- **Input nodes** (forming the **input layer**): accepts the initial data into the system for further processing
- **Weights and biases**: represents the strength of the connection between units
- **An activation function**

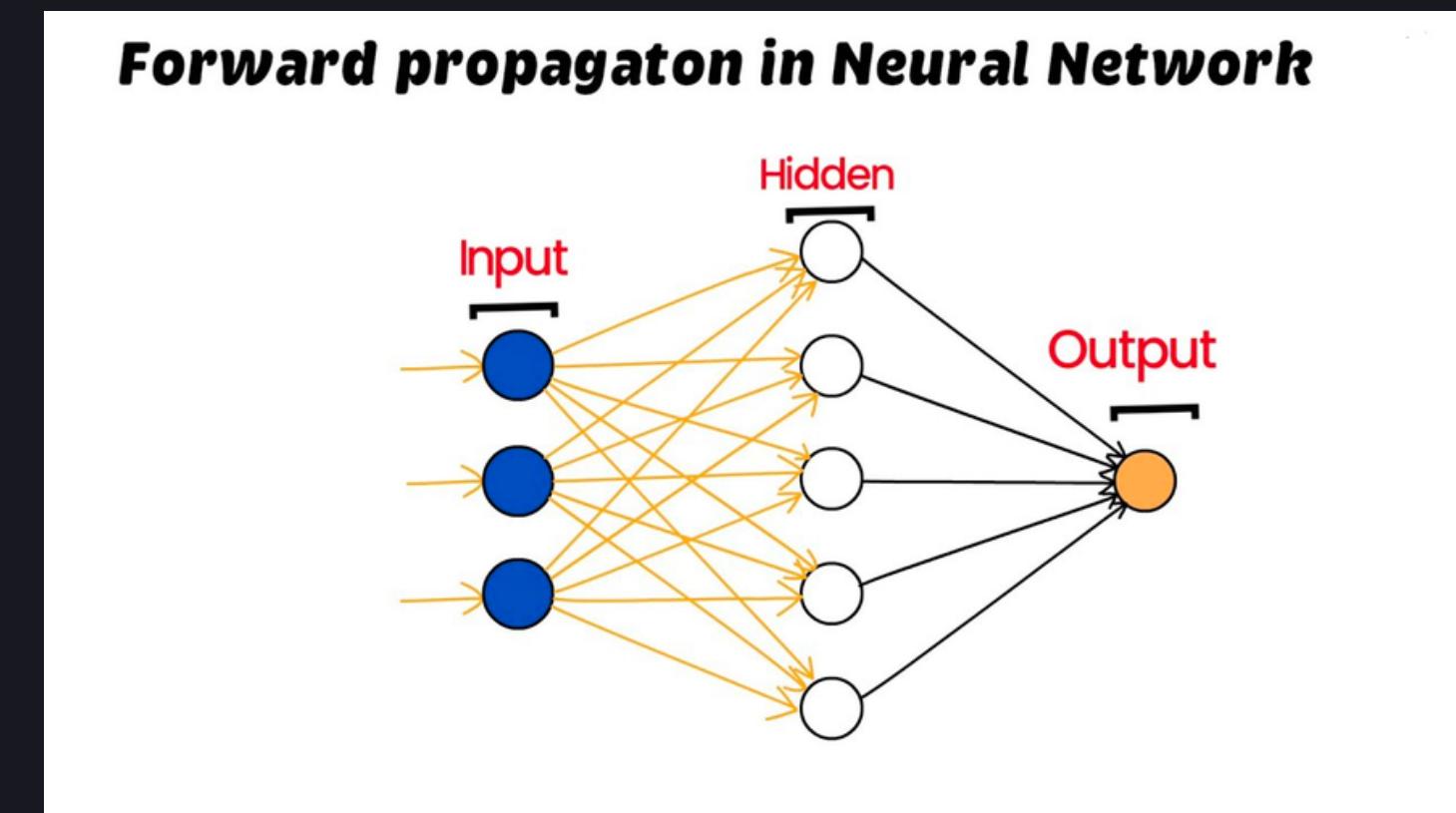


This later inspired the development of the **multi-layer perceptron** model and feedforward neural networks more broadly.

NEURAL NETWORKS

Forward propagation

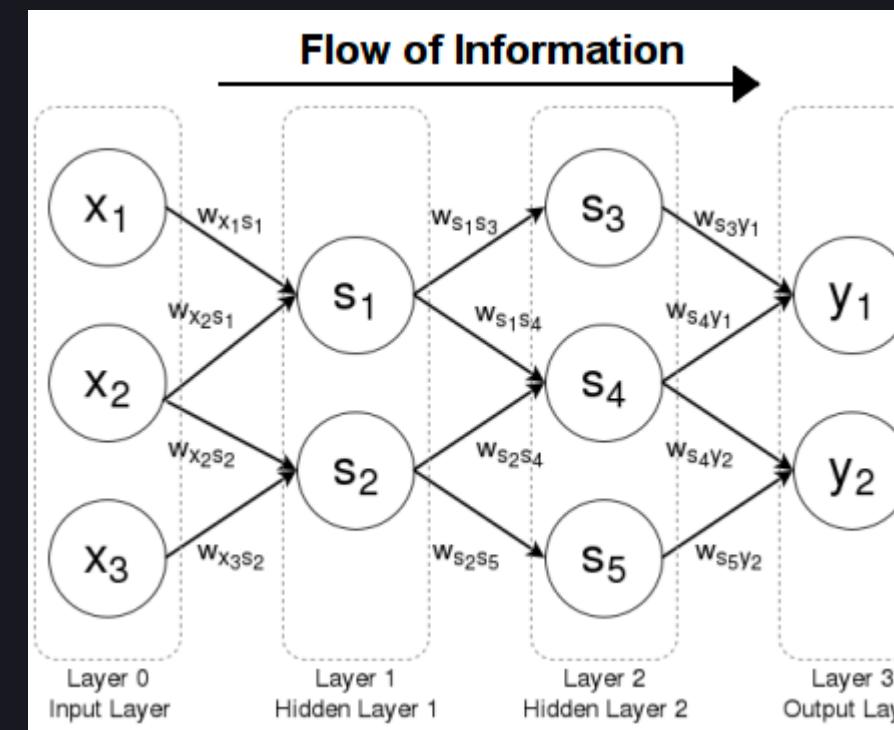
- The input data is fed in the forward direction through the network
- Hidden layer accepts the input data, processes it as per the activation function
- Output is passed to the successive layer



NEURAL NETWORKS

Feed-forward neural networks

- To generate output, the input data is fed in through the forward direction only.
- The data should not flow in reverse direction – it would form a cycle and the output could never be generated
- Such network configurations are known as feed-forward networks.



NEURAL NETWORKS

Feed-forward neural networks

The Two Steps:

- **Pre-activation:** it is a weighted sum of inputs, i.e. the linear transformation of weights w.r.t to the inputs available. Based on this aggregated sum and activation function, the neuron makes a decision as to how much information to pass further or not.
- **Activation:** the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function which introduces non-linearities into the network. There are four commonly used and popular activation functions: sigmoid, hyperbolic tangent (tanh, sometimes pronounced 'tanch'), ReLU and softmax.

Theory

NEURAL NETWORKS

```
# define the training procedure
# i.e. one step of gradient descent
# there are lots of steps
# so we encapsulate it in a function
# Note: inputs and labels are torch tensors
def train(model, loss, optimizer, inputs, labels):
    # https://discuss.pytorch.org/t/why-is-it-recommended-to-wrap-your-data-with-variable
    inputs = Variable(inputs, requires_grad=False)
    labels = Variable(labels, requires_grad=False)

    # Reset gradient
    # https://discuss.pytorch.org/t/why-do-we-need-to-set-the-gradients-manually-to-zero-
    optimizer.zero_grad()

    # Forward
    logits = model.forward(inputs)
    output = loss.forward(logits, labels)

    # Backward
    output.backward()

    # Update parameters
    optimizer.step()
```

NEURAL NETWORKS

Loss calculation

Mean Squared Error

$$\text{loss} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Cross-entropy

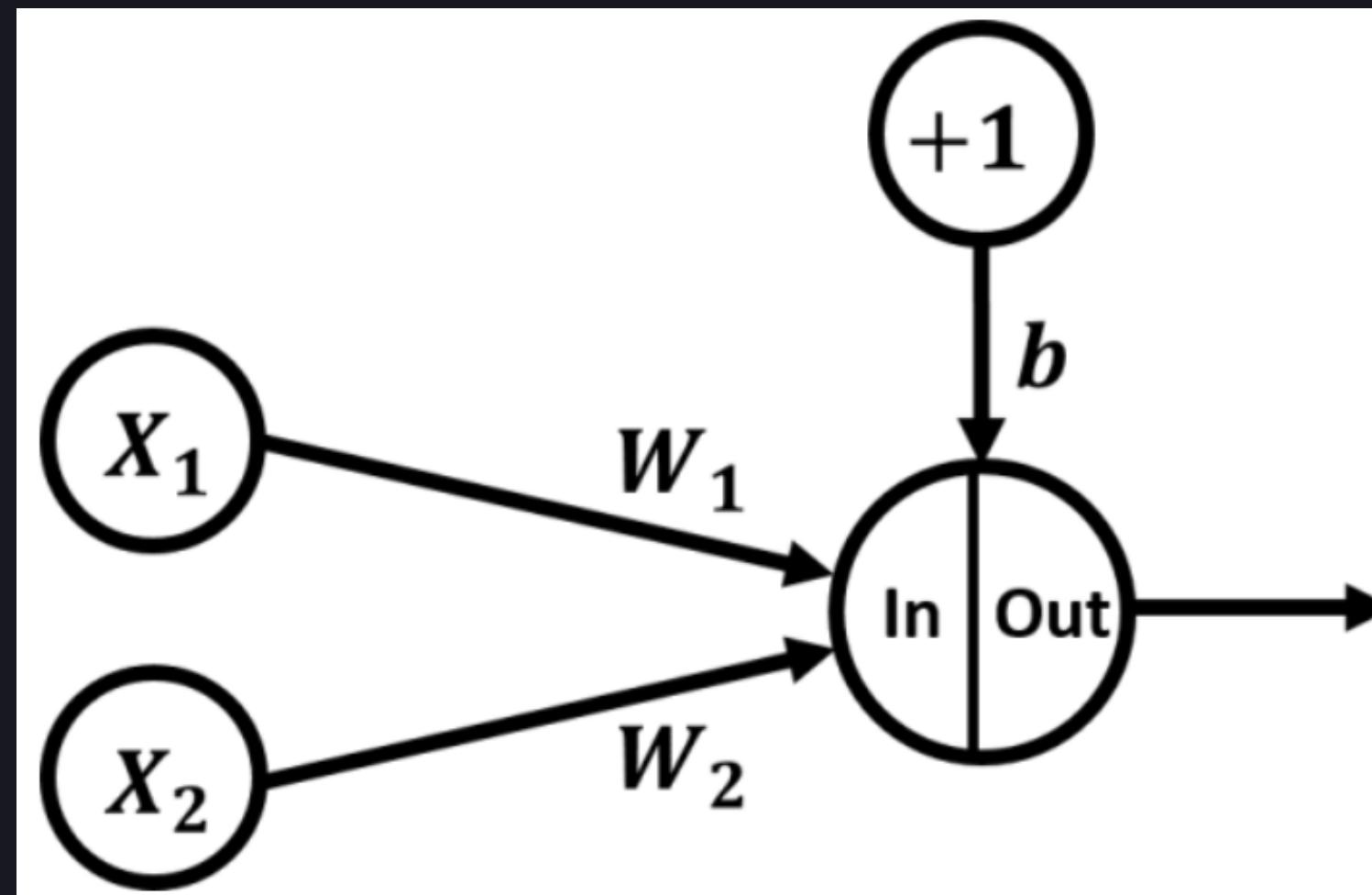
$$\text{loss} = - \sum_{i \in C} y_i \log(\hat{y}_i)$$

Binary cross-entropy

$$\text{loss} = -y_1 \log(\hat{y}_1) - (1 - y_1) \log(1 - \hat{y}_1)$$

NEURAL NETWORKS

Backpropagation



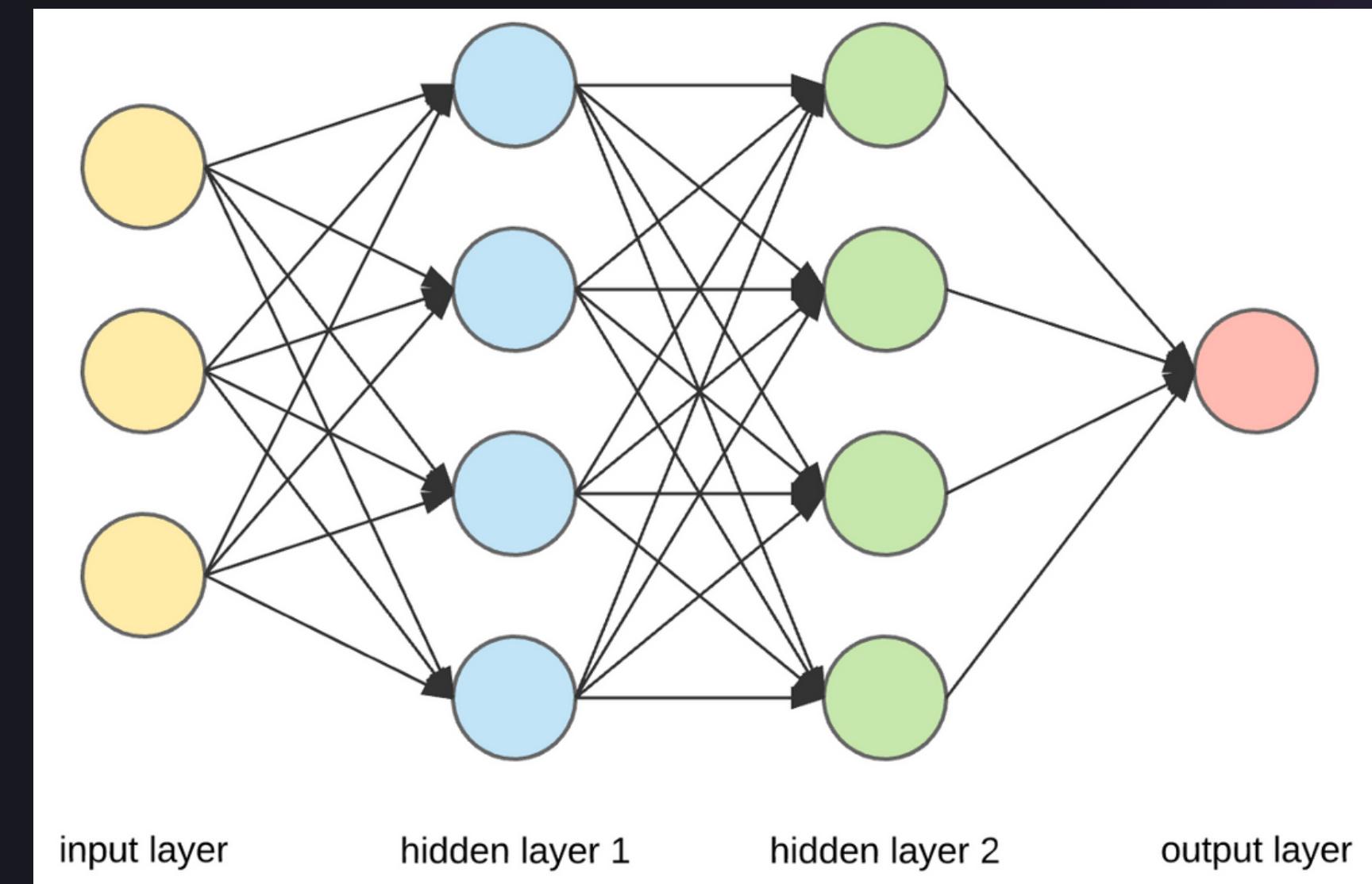
- Taking the loss derivative with respect to each weight and bias using the chain rule
- Updating weights (tempered by a learning rate)

$$\frac{\partial \text{loss}}{\partial w_1} = \frac{\partial \text{loss}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w_1}$$
$$w_{1 \text{ new}} = w_1 - \text{LR} \frac{\partial \text{loss}}{\partial w_1}$$

NEURAL NETWORKS

Multiple layers

- The output from one perceptron becomes the input of another perceptron in the next layer along
- This allows for more complex problems to be tackled and forms the basis of **deep learning**



IT'S TIME FOR YOUR
FIRST CHALLENGE



Putting it all into practice

DOXA

*A platform for hosting
engaging artificial
intelligence competitions*

We are thrilled to launch the first
UCLAIS tutorial series challenge on

doxaai.com

This is a great opportunity to put
what you have learned into practice!

DOXA

Home

Competitions

Profile



DOXA

A powerful platform for hosting engaging artificial intelligence competitions.

[Discover competitions](#) · [Join our Discord community](#)

330+ users · 2,230+ submissions · 3,400+ evaluations

Featured competitions



Climate Hack.AI 2023

Climate Hack.AI is a joint initiative between the AI communities of 25 world-leading universities across the UK, US and Canada, singularly focused on making a direct climate impact through machine learning. Next year, participants will be challenged to develop cutting-edge models that could support the National Grid Electricity System Operator in reducing UK carbon emissions by up to 100,000 tonnes a year.



UCL AI Society Tutorial Series

The [UCL Artificial Intelligence Society](#) hosts a weekly series of tutorials to help the UCL student community engage with machine learning, covering everything from the fundamentals of machine learning to the innovative ideas underpinning computer vision, natural language processing and reinforcement learning.

Our custom platform

DOXA is a powerful and highly customisable platform for running engaging online artificial intelligence competitions. From hosting computationally intensive high-impact challenges, such as satellite imagery nowcasting, to fun multiplayer game tournaments, such as in Ultimate Tic-tac-toe, DOXA has been designed from the ground up to be performant, robust and scalable.

Built atop a React-Rust technology stack, DOXA evaluated over 2,200 competition submissions from 300+ participants from across the UK and North America for Climate Hack.AI 2022, seeing everyone from seasoned researchers to enthusiastic novices new to computer vision engage with machine learning for the social good.

Find out more about [how DOXA works](#).

[About DOXA](#)

Copyright © 2022 DOXA

BRAIN STROKE PREDICTION

Area of Application: Disease Diagnosis

Challenge

Predict whether or not someone has suffered a brain stroke using the machine learning techniques you have learned over the past few tutorial lectures



Training Set: 4500 entries with 10 input features and one binary output label

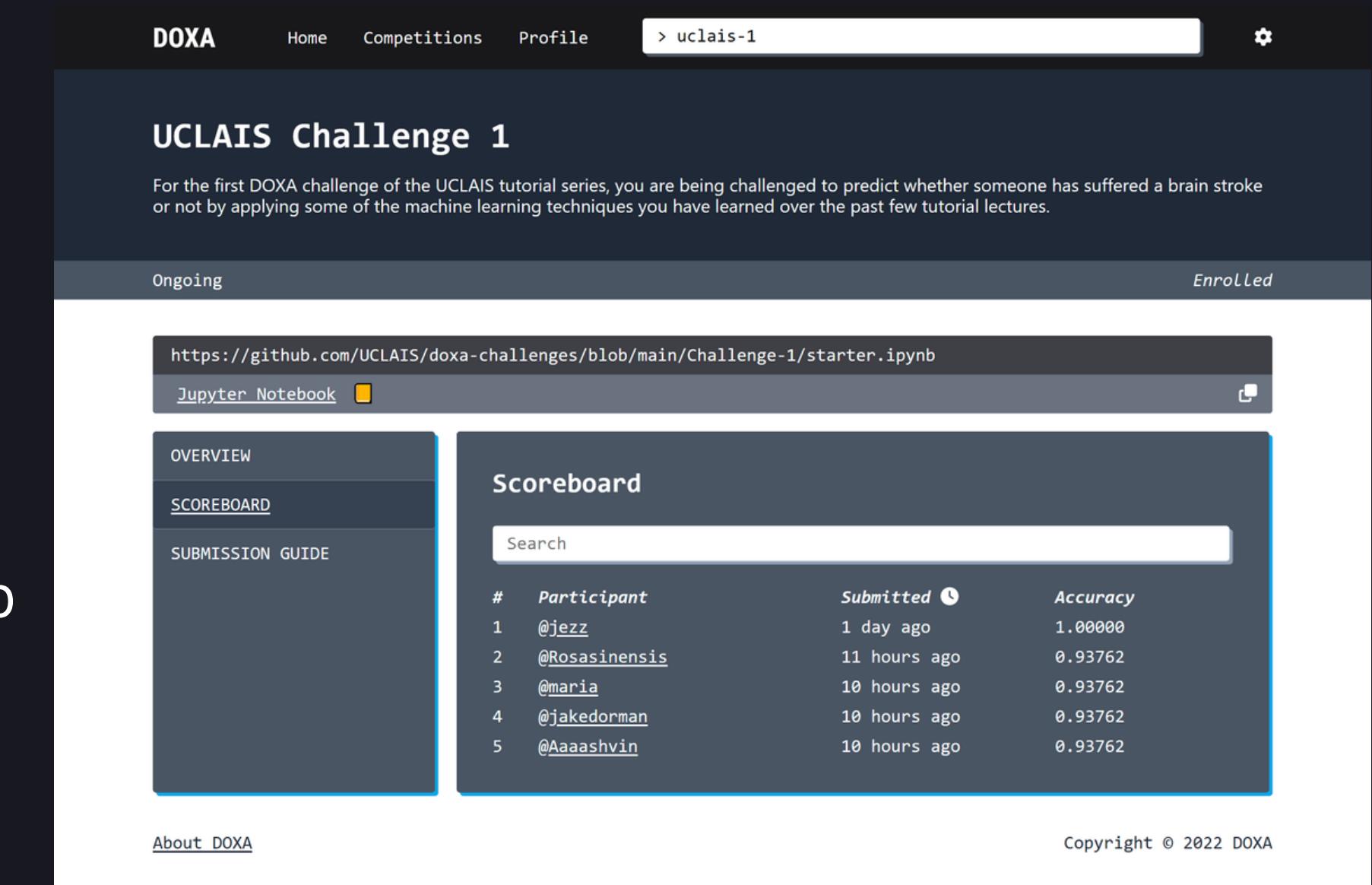
Test Set: 481 entries containing only the 10 input features without the output label

FEATURES

Gender, age, hypertension, heart disease, BMI, marital status, work type, etc...

HOW TO TAKE PART

1. Sign up for an account on doxaai.com
2. Enrol for the competition at
<https://doxaai.com/competition/uclais-1>
3. Open the Jupyter notebook in Google Colab
(<https://colab.research.google.com/>)
4. Run all the cells (making sure to approve access for the DOXA CLI submission tool)
5. Find yourself on the DOXA scoreboard!
6. Improve on the model you were provided with using some of the ideas you have learned over the past few weeks.



The screenshot shows the DOXA competition interface for "UCLAIS Challenge 1". The top navigation bar includes links for DOXA, Home, Competitions, Profile, and a user dropdown. The main title "UCLAIS Challenge 1" is displayed, with a subtitle explaining the challenge: "For the first DOXA challenge of the UCLAIS tutorial series, you are being challenged to predict whether someone has suffered a brain stroke or not by applying some of the machine learning techniques you have learned over the past few tutorial lectures." Below the title, there are two tabs: "Ongoing" and "Enrolled", with "Ongoing" selected. A link to the Jupyter Notebook (<https://github.com/UCLAIS/doxa-challenges/blob/main/Challenge-1/starter.ipynb>) is shown. The left sidebar has three sections: OVERVIEW, SCOREBOARD (which is currently active), and SUBMISSION GUIDE. The right panel displays a "Scoreboard" table with the following data:

#	Participant	Submitted	Accuracy
1	@jezz	1 day ago	1.00000
2	@Rosasinensis	11 hours ago	0.93762
3	@maria	10 hours ago	0.93762
4	@jakedorman	10 hours ago	0.93762
5	@Aaaashvin	10 hours ago	0.93762

At the bottom of the page are links for "About DOXA" and "Copyright © 2022 DOXA".

DOXA: <https://doxaai.com/>

GitHub: <https://github.com/UCLAIS/doxa-challenges/tree/main/Challenge-1>

Notebook: <https://github.com/UCLAIS/doxa-challenges/blob/main/Challenge-1/starter.ipynb>

A DEMO ☀

QUESTIONS?



SEE YOU NEXT TIME