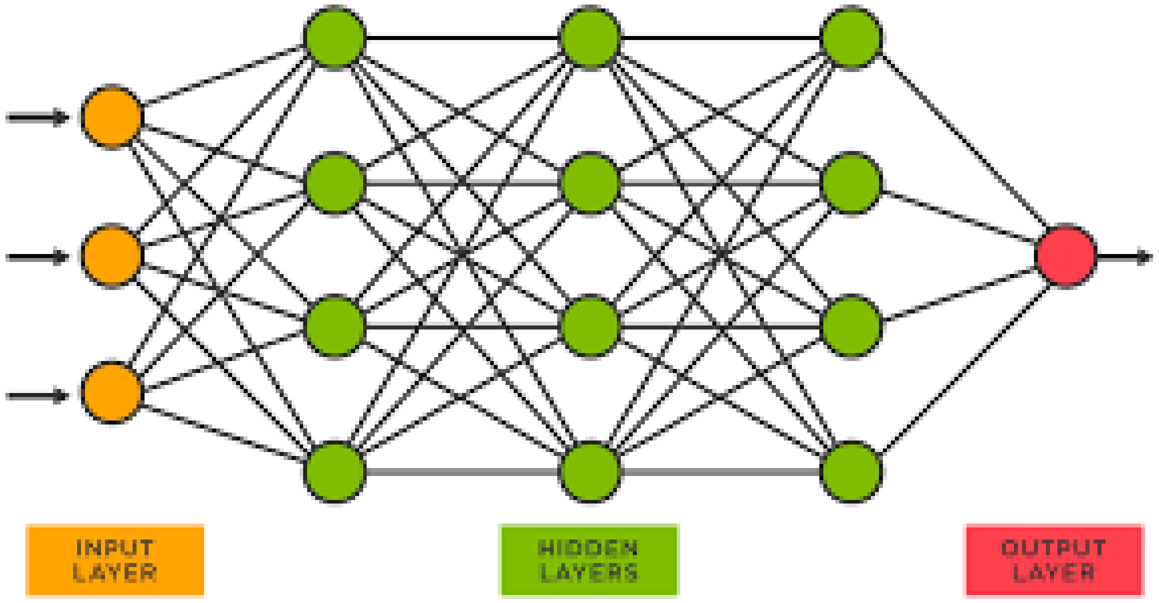
MIS4311
Machine Learning Applications

Fall 2025

Lecture #7

Neural Networks

A neural network is a method in AI that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that **uses interconnected nodes or neurons in a layered structure** that resembles the human brain.



Neural Networks

What are neural networks used for?

- Medical diagnosis by medical image classification
- Targeted marketing by social network filtering and behavioral data analysis
- Financial predictions by processing historical data of financial instruments
- Electrical load and energy demand forecasting
- Process and quality control
- Chemical compound identification

Neural Networks Architecture

A basic neural network has interconnected artificial neurons in 3 layers:

Input Layer

Information from the outside world enters the artificial neural network from the input layer. Input nodes **process the data, analyze or categorize it**, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers. **Artificial neural networks can have a large number of hidden layers.** Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output Layer

The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance**, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0.** However, if we have **a multi-class classification** problem, the output layer might consist of **more than one output node**.

Deep Neural Network Architecture

Deep neural networks, or deep learning networks, have **several hidden layers with millions of artificial neurons linked together.**

A number, called **weight**, **represents the connections between one node and another.** The weight is a positive number if one node excites another, or negative if one node suppresses the other.

Nodes with **higher weight values have more influence on the other nodes.**

Theoretically, deep neural networks can **map any input type to any output type.** However, they also need much more training as compared to other machine learning methods. They need millions of examples of training data rather than perhaps the hundreds or thousands that a simpler network might need.

Types of Deep Neural Networks

Feedforward neural networks: process data in **one direction**, from the input node to the output node. Every node in one layer is connected to every node in the next layer.

Backpropagation algorithm:

The data flowing from the input node to the output node through many different paths in the neural network. **Only one path is the correct one that maps the input node to the correct output node**. To find this path, the neural network uses a feedback loop, which works as follows:

- Each node **makes a guess** about the next node in the path.
- It checks if the guess was correct. Nodes assign **higher weight values to paths that lead to more correct guesses and lower weight values to node paths that lead to incorrect guesses.**
- For the next data point, the nodes make a new prediction using the higher weight paths and then repeat Step 1.

Types of Deep Neural Networks

Convolutional neural networks:

Convolution is a mathematical **combination of two functions** which involves **multiplying the value of one function at a given point with the value of another.**

The hidden layers in convolutional neural **networks perform specific mathematical functions, like summarizing or filtering, called convolutions.**

They are very useful for image classification because **they can extract relevant features from images that are useful for image recognition and classification.** The new form is easier to process without losing features that are critical for making a good prediction.

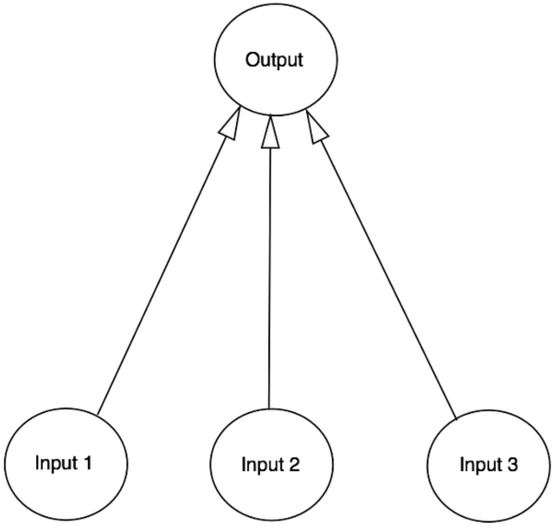
Each hidden layer extracts and processes different image features, like edges, color, and depth.

Neural Networks Architecture

The human brain consists of 100 billion cells called **neurons**, **connected together by synapses**. If **sufficient synaptic inputs** to a neuron fire, that neuron will also fire. We call this process **“thinking”.**

Model a single neuron, with three inputs and one output.

Train the neuron to solve the problem below. The first four examples are training set, the last line is to be predicted.



	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0

New situation	1	0	0	**?**

Neural Networks Architecture

Training process:

We will give each input a **weight**, which can be a positive or negative number.

An input with a **large** positive **weight** or a large negative weight, will have a **strong effect** on the neuron's output.

1. Take the inputs from a training set example, adjust them by the weights, and pass them through a special formula to **calculate the neuron's output.**

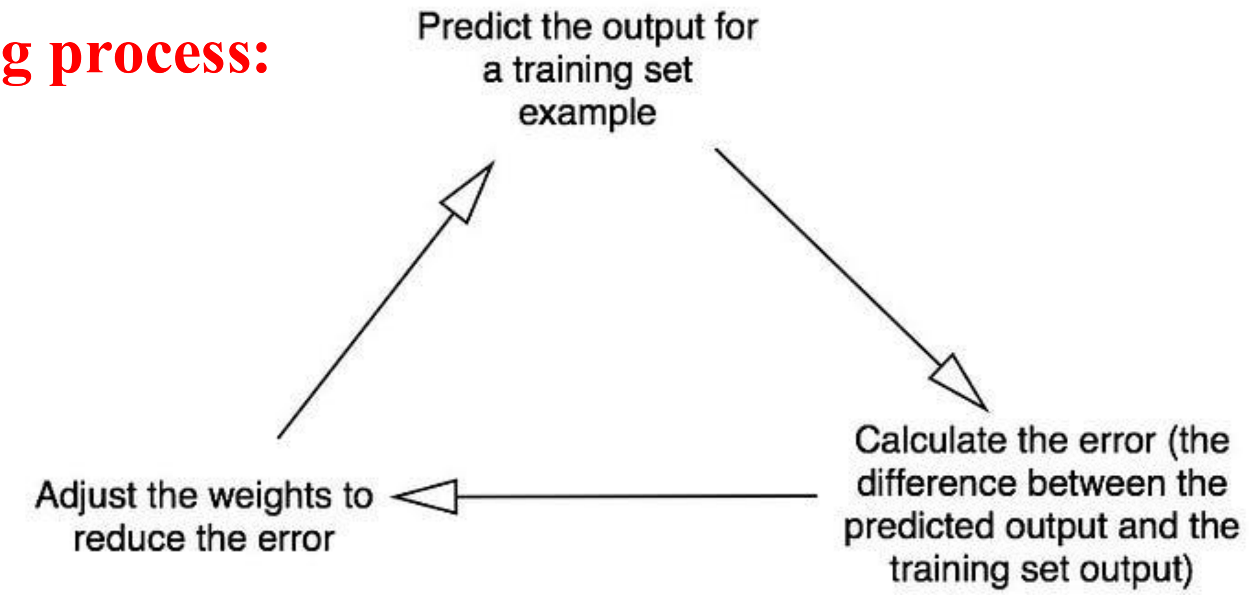
2. Calculate the error, which is the difference between the neuron's output and the desired output in the training set example.

3. Depending on the direction of the error, **adjust the weights slightly.**

4. Repeat this process many times (e.g., 10000).

Neural Networks Architecture

Training process:



Predict the output for a training set example

Calculate the error (the difference between the predicted output and the training set output)

Adjust the weights to reduce the error

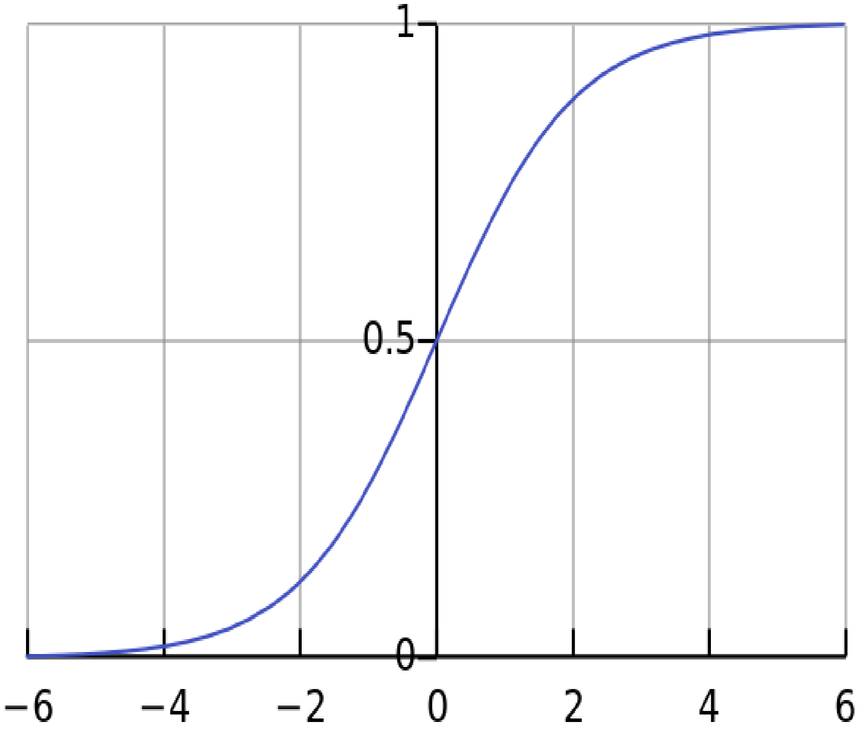
Formula for calculating the neuron’s output:

$$\sum weight_i \cdot input_i = weight1 \cdot input1 + weight2 \cdot input2 + weight3 \cdot input3$$

Neural Networks Architecture

Training process: Normalize output of neuron

Normalize the weighted sum of the neuron's inputs by Sigmoid function, so the result will be between 0 and 1.



$$\frac{1}{1+e^{-x}}$$

$$\text{Output of neuron} = \frac{1}{1+e^{-(\sum weight_i input_i)}}$$

Neural Networks Architecture

Training process: Adjusting the weights

Use the “Error Weighted Derivative” formula:

$$\text{Adjust weights by} = error \cdot input \cdot SigmoidCurveGradient(output)$$

The gradient of the Sigmoid curve, can be found by taking the derivative:

$$\text{SigmoidCurveGradient(output)} = output \cdot (1 - output)$$

Thus the final formula for adjusting the weights is:

$$\text{Adjust weights by} = error \cdot input \cdot output \cdot (1 - output)$$

where

$$\text{Output of neuron} = \frac{1}{1 + e^{-(\sum weight_i input_i)}}$$

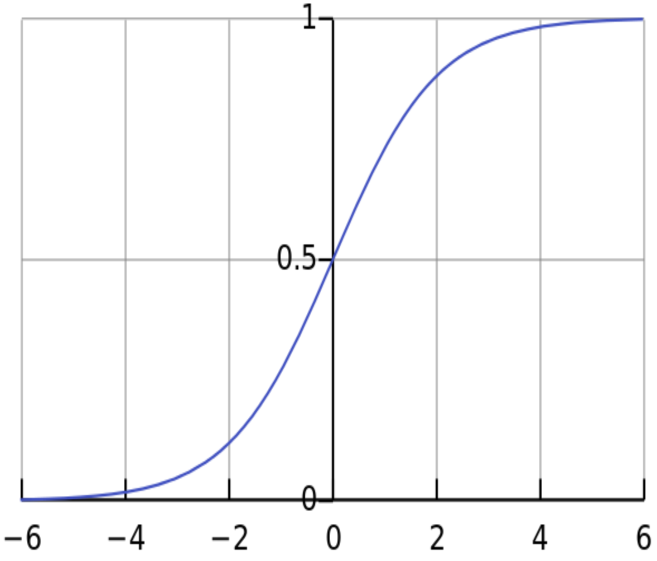
Neural Networks Architecture

Training process: Outline

The final formula for adjusting the weights is:

$$\text{Adjust weights by} = error \cdot input \cdot output \cdot (1 - output)$$

1. Use the Sigmoid curve to calculate the output of the neuron.
2. If the output is a large positive or negative number, it signifies the neuron was quite confident one way or another.
3. **At large numbers, the Sigmoid curve has a shallow gradient.**
4. If the neuron is confident that the existing weight is correct, it doesn't want to adjust it very much. Multiplying by the Sigmoid curve gradient achieves this.



Artifcial Neural Networks with Python

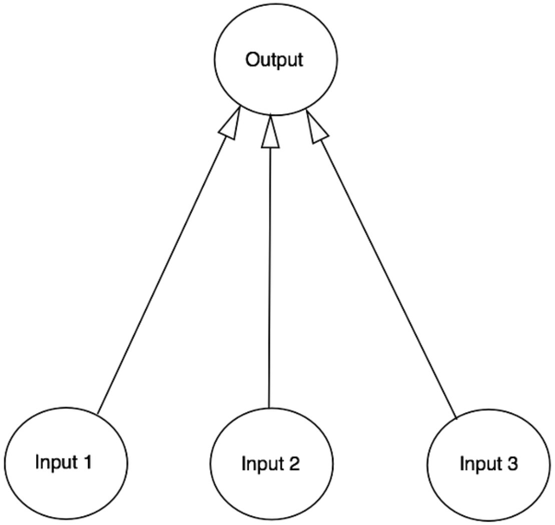
Install packages:
conda install tensorflow
conda install keras

Keras is a powerful and easy-to-use free open source Python library for **developing and evaluating deep learning models**.

It is part of the Tensorflow library allows you to **define and train neural network models.**

Artifcial Neural Networks with Python

Application 1: Train the neuron to solve the problem below. The first four examples are training set, the last line is to be predicted.



	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0

New situation	1	0	0	**?**

Artifcial Neural Networks with Python

Application 2: Use a Neural Network to Predict Taxi Fares using data from the New York City Taxi & Limousine Commission to predict taxi fares. We’ll use a neural **network as a regression model** to make the predictions.

Download *taxi-fares.csv* from Marmara cloud system.

The data features:

1. key (time as a unique data)
2. fare_amount → output variable
3. pickup_datetime
4. pickup_longitude
5. pickup_latitude
6. dropoff_longitude
7. dropoff_latitude
8. passenger_count

Next Week

- Midterm Exam

Thank you for your participation ☺