Introduction to Deep learning

Graduate School Orléans Numérique

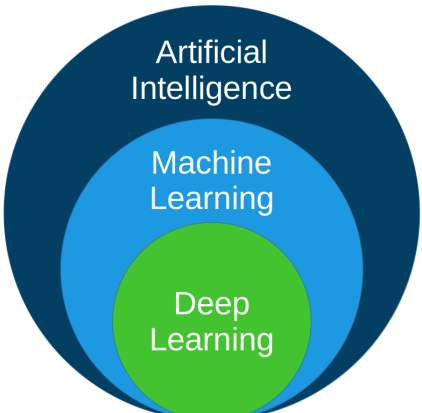
Daniel Förster
UFR Sciences et Techniques Université d'Orléans

Agenda

- Introduction: What is Deep Learning?
- Datasets and best practices
- Applications and first real-world example

`gson-deeplearning.surge.sh`

What is Deep Learning?

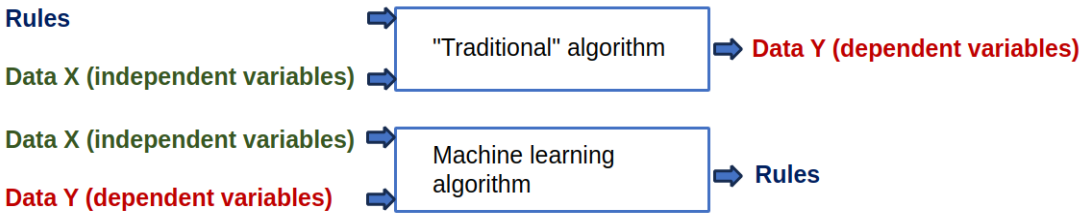


Artificial intelligence:
A program that can sense, reason, act, and adapt

Machine learning:
Algorithms that learn from data

Deep learning:
Machine learning using multi-layered artificial neural networks

Supervised learning



- Requires large amounts of labelled data
- Can be sometimes obtained through simulation

Place of machine learning in science

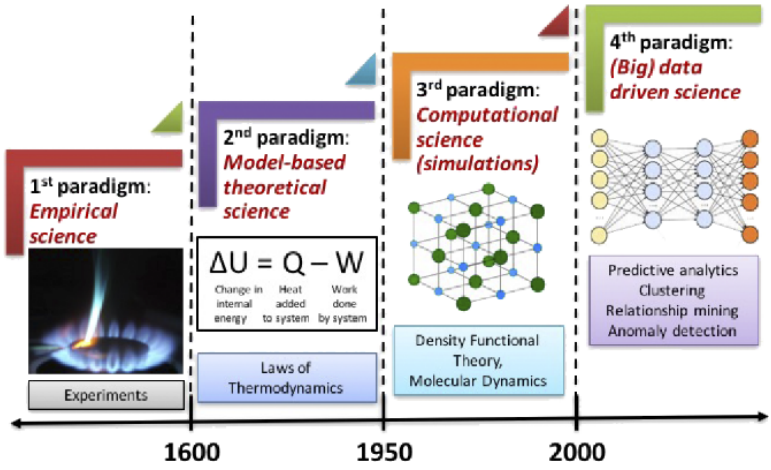


Image source [2]

Deep learning history

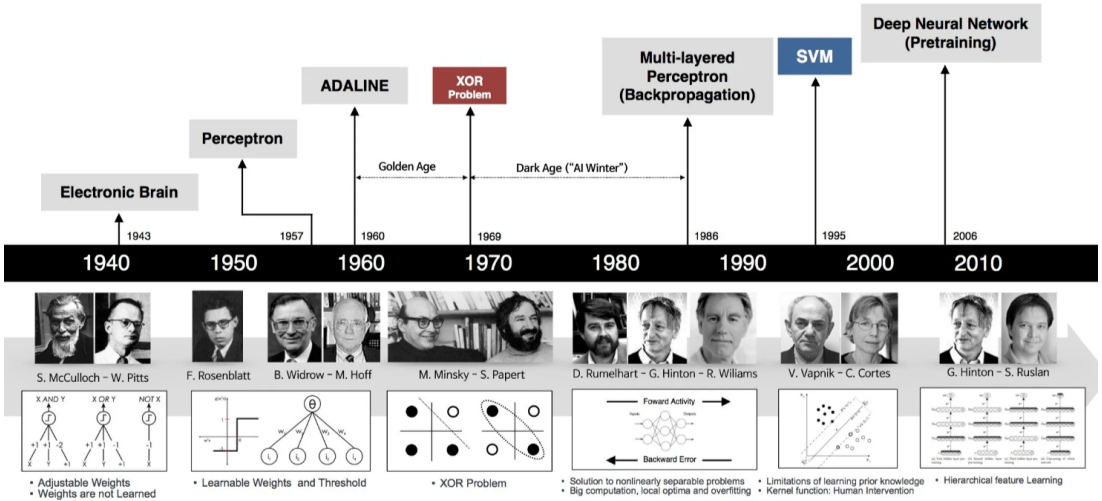
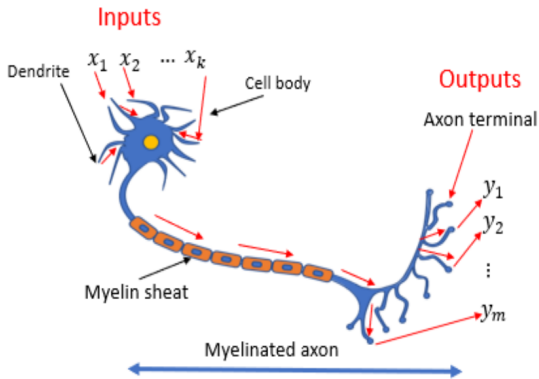


Image source [1]

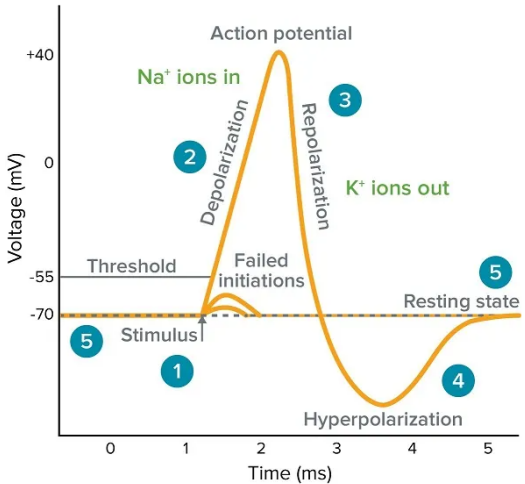
Biological neuron



$\rightarrow$ Inspiration for artificial neural networks

Image source [3]

Action potentials

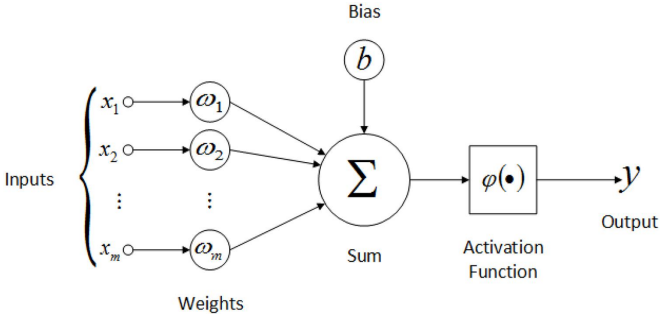


All-or-none principle: Action potential is either generated or not according to a voltage threshold

Frequency of action potentials encodes signal intensity

Image source [4]

Anatomy of an Artificial Neuron (perceptron)



Output y:

$$y = \phi\left(\sum_{i=1}^{m} x_i w_i + b\right)$$

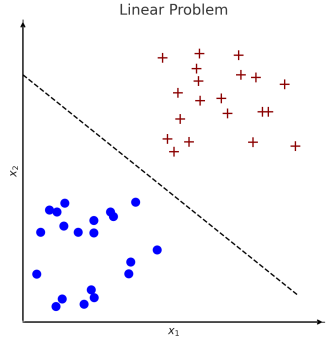
with m number of inputs, w_i "weights", b "bias", and ϕ "activation function".

Perceptron: activation function is Heaviside step function Θ

$\rightarrow$ **Linear classifier:** impossibility to model the `xor` function

Image source [5]

Perceptron in 2D



How can the weights w_i be obtained?

Training set:
(x_1^i, x_2^i) with labels d^i

Training by:

$$w_j \leftarrow w_j + l\left(d^i - y^i\right) x_j^i$$

where l is the "learning rate"

Output:

$$y = \Theta\left(w_1 x_1 + w_2 x_2 + b\right)$$

with separatrix: $x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \rightarrow b = -1$

Building a Network: Layers

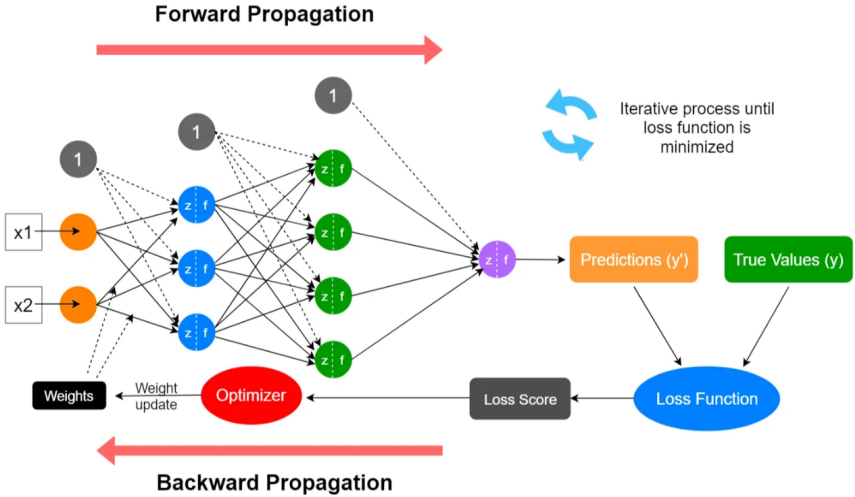


Image source [8]

Dataset

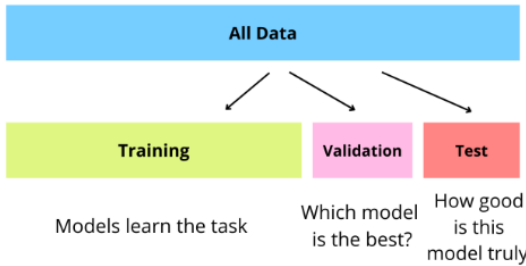
Examples from which the system learns: For each input x, there is an output (or label) y. Both input and output are usually vectors (or matrices); the output can be a single number as well.
The data set should be:

- representative of the problem, diverse
- balanced (distribution over the input and output data, no duplicates)
- outlier-free
- normalized (e.g. input and output values between zero and one)
- accurate
- sufficiently large

Sometimes it is possible to **simulate** data, making it easier to increase the size and quality of the data set.

Training, Validation, and Test Sets

Dividing a dataset into subsets to train, tune, and evaluate the deep learning model



Common split ratios are 60-20-20 or 70-15-15 for training-validation-test.

Data leakage happens when information from the validation/test sets unintentionally gets used during training. $\rightarrow$ Overly optimistic performance estimate.

Image source [1]

Introduction to Supervised Learning

- **Supervised Learning** involves learning a mapping from inputs to outputs using labeled datasets.
- Two primary tasks:
  - **Regression**: Predicting continuous numerical values.
  - **Classification**: Assigning inputs to discrete categories.
- Applications span various fields like finance, healthcare, and technology.

Understanding the differences between regression and classification is crucial for selecting appropriate models.

Example problems

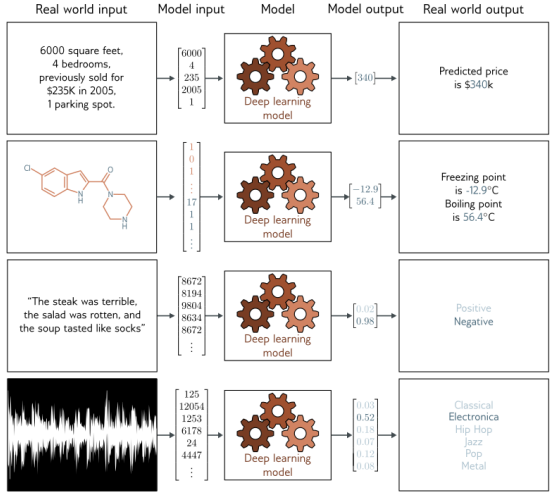


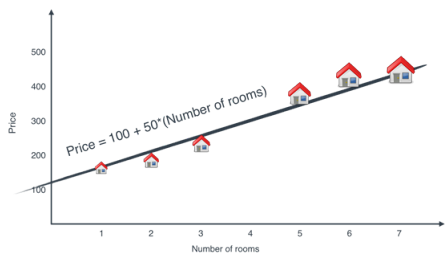
Image source [2]

Regression

- **Objective**: Predict continuous numerical values based on input features.
- **Examples**:
  - House price prediction using features like size, location, and number of bedrooms.
  - Forecasting stock prices or weather conditions.
- **Model Representation**:

$$\hat{Y} = f(X; \theta)$$

where X is the input data, θ represents model parameters, and $\hat{Y}$ is the predicted output.



Output Activation Functions for Regression

- **Linear Activation Function**:

$$f(z) = z$$

  Used when output values are unbounded.

- **ReLU (Rectified Linear Unit)**:

$$f(z) = \max(0, z)$$

  Ensures output is non-negative.

- **Sigmoid Function**:

$$f(z) = \frac{1}{1 + e^{-z}}$$

  Used when output values are constrained between 0 and 1.

Loss Functions in Regression

- **Mean Squared Error (MSE)**:

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

  Measures the average squared difference between actual and predicted values.

- **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

  Computes the average absolute difference.

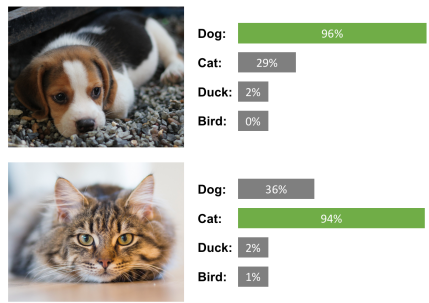
- **Purpose**: Loss functions guide the optimization of model parameters to minimize prediction errors.

Classification

- **Objective**: Assign input data to predefined categories or classes.
- **Types**:
  - **Binary Classification**: Two classes (e.g., spam vs. not spam).
  - **Multi-class Classification**: More than two classes (e.g., digit recognition).
- **Model Representation**:

$$P(y|X) = f(X;\theta)$$

where $P(y|X)$ is the probability of class y given input X.



One-Hot Encoding

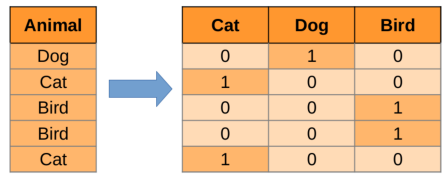
- **Definition**: Converts categorical class labels into binary vectors.
- **Example**:

$$\text{Classes} : \{\text{Cat}, \text{Dog}, \text{Bird}\}$$

$$\text{Cat} \rightarrow [1, 0, 0]$$

$$\text{Dog} \rightarrow [0, 1, 0]$$

$$\text{Bird} \rightarrow [0, 0, 1]$$



Animal
Dog
Cat
Bird
Bird
Cat

Cat	Dog	Bird
0	1	0
1	0	0
0	0	1
0	0	1
1	0	0

- **Purpose**: categorical data

Activation Functions in Classification

- **Sigmoid Function** (Binary Classification):

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

  Maps any real-valued number into the range (0, 1).

- **Softmax Function** (Multi-class Classification):

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

  Produces a probability distribution over K classes.

- **Properties**:
  - Outputs sum to 1.
  - Each output is between 0 and 1.

Loss Functions in Classification

- **Binary Cross-Entropy Loss**:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

  Used for binary classification tasks.

- **Categorical Cross-Entropy Loss**:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k})$$

  Applicable to multi-class classification.

- **Interpretation**:
  - Measures the dissimilarity between true labels and predicted probabilities.
  - Encourages the model to assign high probabilities to the correct classes.

Building a Network: Layers (again)

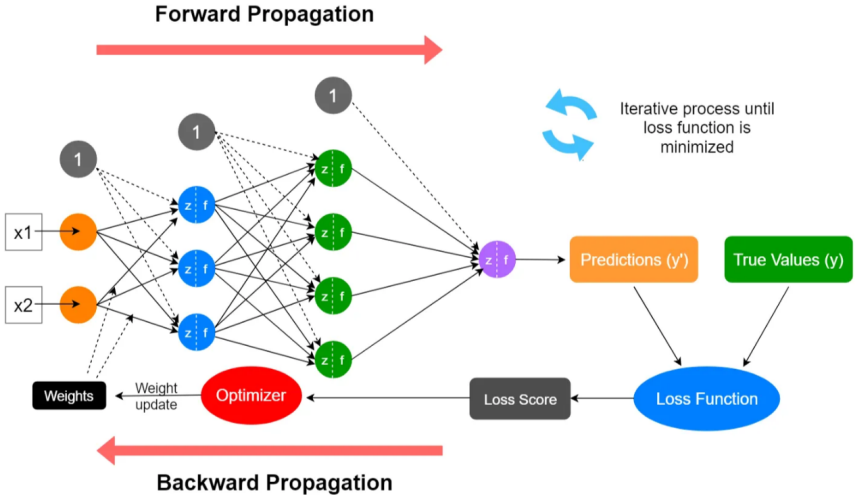


Image source [8]

Application: MNIST dataset

- 70000 labelled images
- each image: 28 28 pixels

Objective: Sort images into a finite number of predefined categories

Image source [9]

First attempt using tensorflow

Things to consider:

- Images need to be reshaped into vectors
- Grayscale values should be between 0 and 1
- Calculate loss function as the categorical crossentropy

$\rightarrow$ Demo