INTRODUCTION TO MACHINE LEARNING IN PHYSICS

ASHRAF MOHAMED

Research fellow

The University of Texas at Austin
Department of Chemistry
College of Natural Sciences

2ND ARPS SUMMER SCHOOL OF ADVANCED PHYSICS

ZEWAIL CITY OF SCIENCE TECHNOLOGY, EGYPT

AUGUST 28, 2024

Arab Physical Society (ArPS)
الجمعية العربية للفيزياء

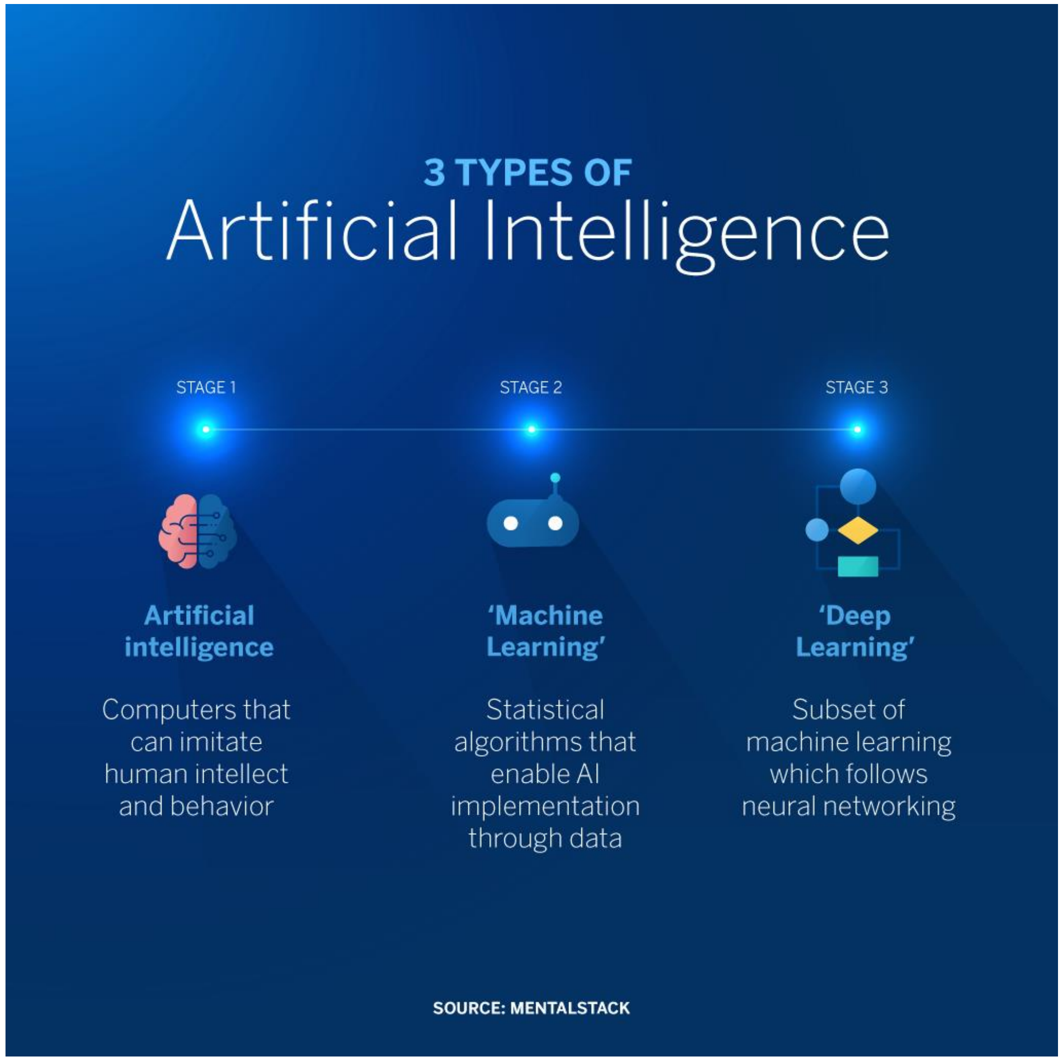
Agenda

Introduction

Why Machine Learning?

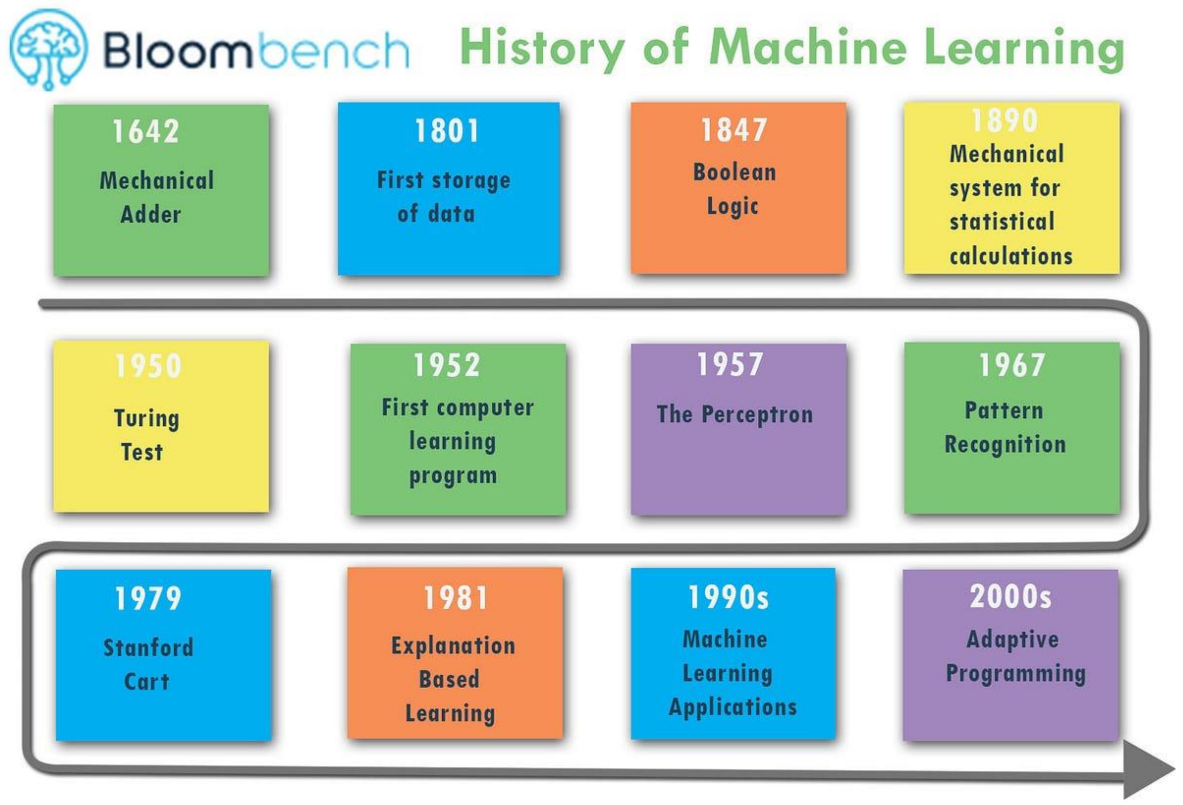
And what?

- Computers can learn from data and make decisions without being explicitly programmed to:
  - Solve complex problems,
  - Discover new insights,
  - Create innovative products and services,
  - Enhancing health care, education, entertainment, security, and more.
- Machines can perform tasks that typically require human intelligence
- ML is a subset of artificial intelligence that focuses on analyzing and interpreting patterns and structures in data to enable learning, reasoning, and decision-making
- ML algorithms can learn from data


3 TYPES OF
Artificial Intelligence
STAGE 1
STAGE 2
STAGE 3
Artificial intelligence
Computers that can imitate human intellect and behavior
'Machine Learning'
Statistical algorithms that enable AI implementation through data
'Deep Learning'
Subset of machine learning which follows neural networking
SOURCE: MENTALSTACK

Historical Context

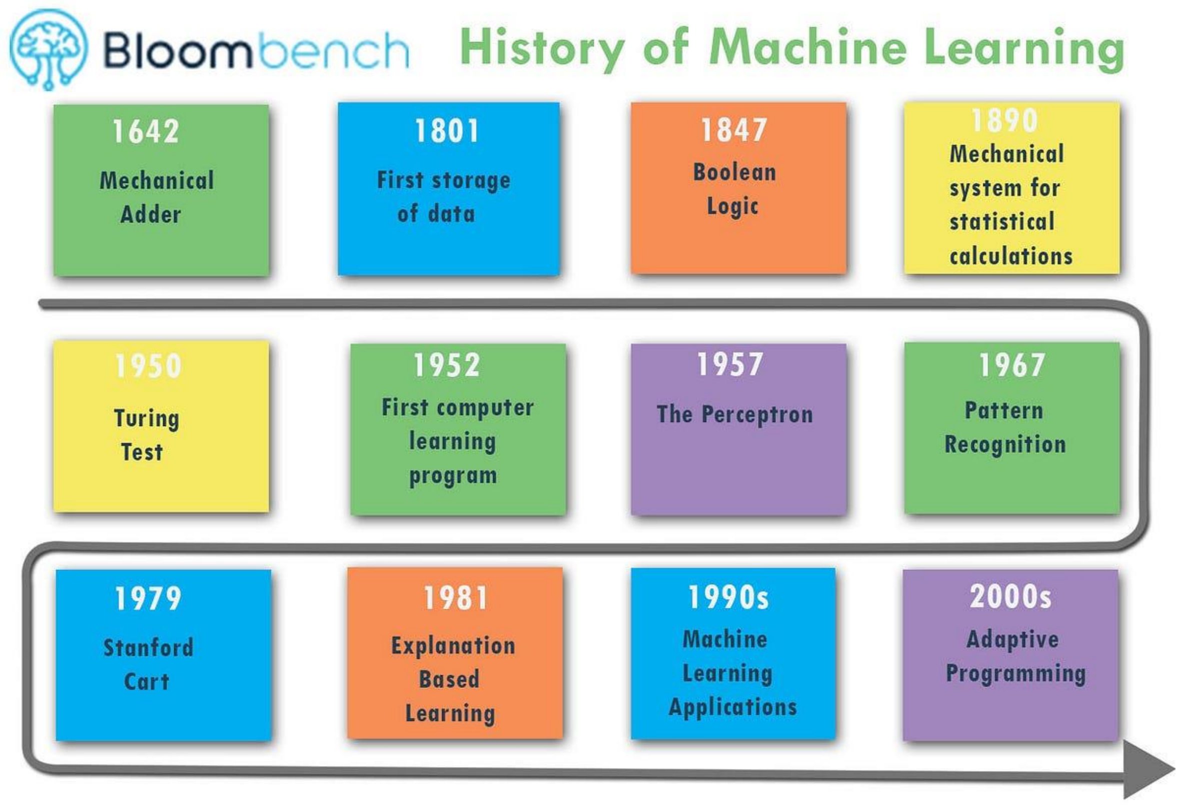
The Evolution of Machine Learning


Bloombench
History of Machine Learning
1642 Mechanical Adder
1801 First storage of data
1847 Boolean Logic
1890 Mechanical system for statistical calculations
1950 Turing Test
1952 First computer learning program
1957 The Perceptron
1967 Pattern Recognition
1979 Stanford Cart
1981 Explanation Based Learning
1990s Machine Learning Applications
2000s Adaptive Programming

Historical Context

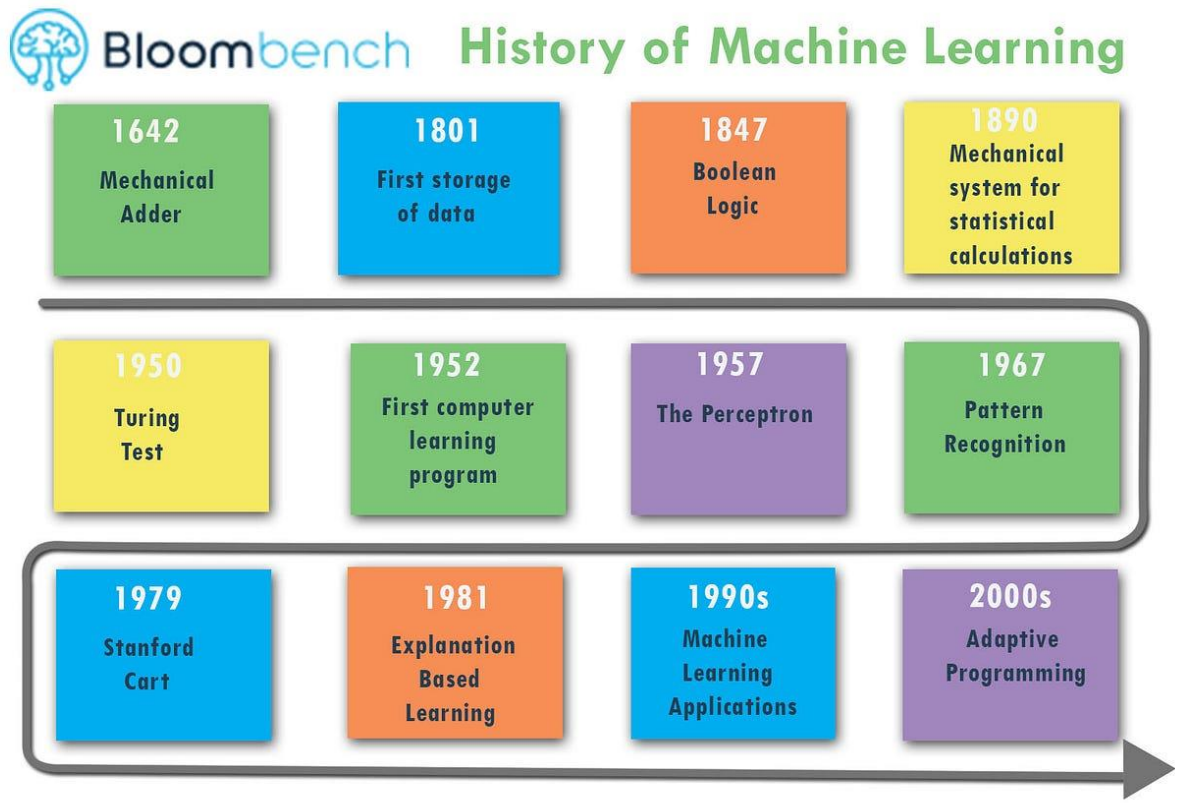
The Evolution of Machine Learning

- **1940s-1950s: The Birth of AI:**
  - Early computer scientists and mathematicians laid the foundation for AI and machine learning concepts.
  - Alan Turing's work on computing and the Turing test marked significant milestones.


Bloombench
History of Machine Learning
1642 Mechanical Adder
1801 First storage of data
1847 Boolean Logic
1890 Mechanical system for statistical calculations
1950 Turing Test
1952 First computer learning program
1957 The Perceptron
1967 Pattern Recognition
1979 Stanford Cart
1981 Explanation Based Learning
1990s Machine Learning Applications
2000s Adaptive Programming

Historical Context

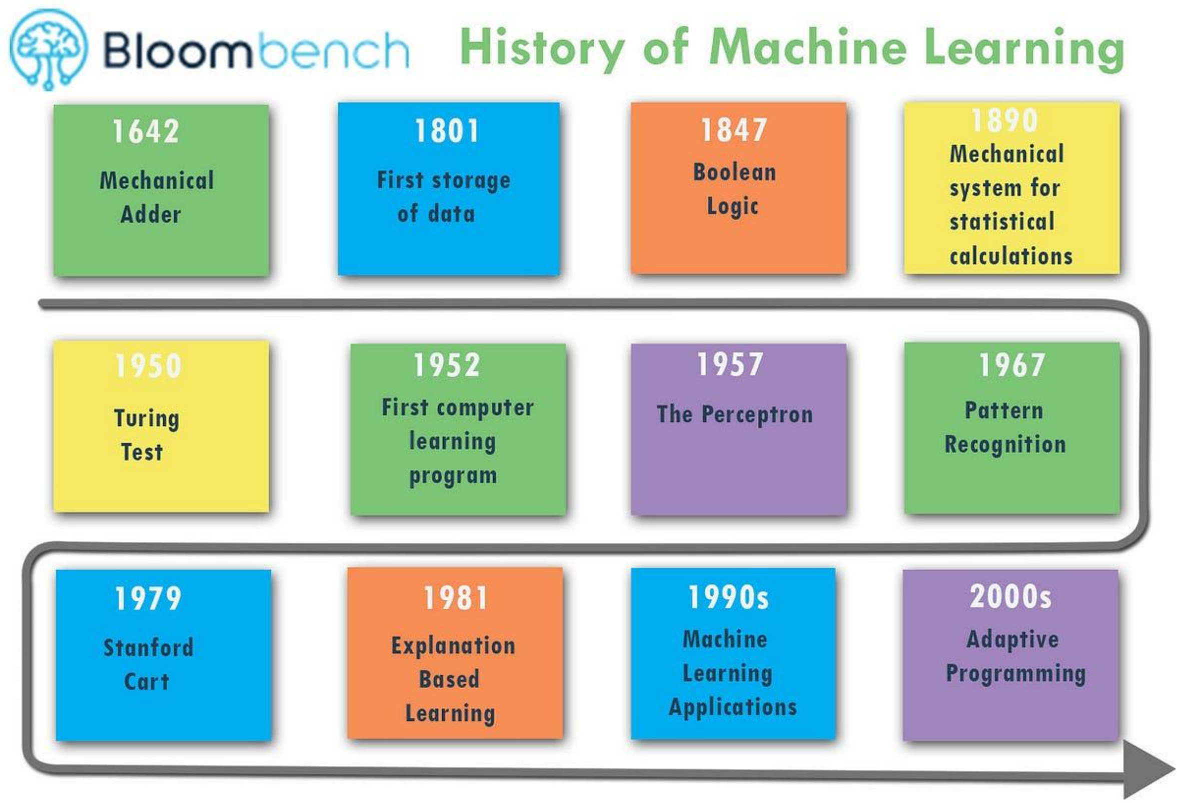
The Evolution of Machine Learning



Historical Context

The Evolution of Machine Learning

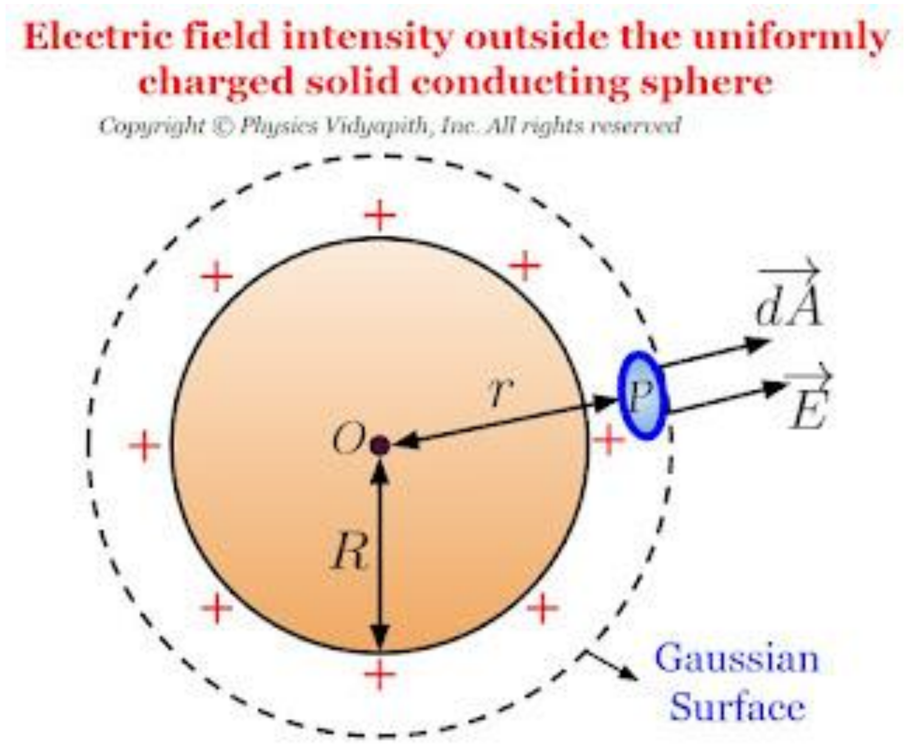
- **1970s-1980s: The First AI Winter:**
  - Disappointing results led to reduced funding and interest in AI and machine learning.
  - Critics believed that AI had overpromised and underdelivered.
- **1990s-Present: The Renaissance:**
  - Advancements in computing power, data availability, and algorithms reignited interest.
  - Machine learning experienced a resurgence, leading to breakthroughs in various applications.


Bloombench
History of Machine Learning
1642 Mechanical Adder
1801 First storage of data
1847 Boolean Logic
1890 Mechanical system for statistical calculations
1950 Turing Test
1952 First computer learning program
1957 The Perceptron
1967 Pattern Recognition
1979 Stanford Cart
1981 Explanation Based Learning
1990s Machine Learning Applications
2000s Adaptive Programming

The power of data

It all about representations

- Data is a crucial component in the field of machine learning
  - Data is a set of observations or measurements
  - Can be used to train a machine-learning model
  - The quality and quantity of data available for training and testing play a significant role in the performance of a machine-learning
- However
  - Data alone is not enough to unleash its power
  - Data needs to be refined, processed, and analyzed using scientific methods and algorithms
  - This is where data science and artificial intelligence come in
- The power of data comes from its representation
  - For Physics students: think of the Gauss law, Dirac notation


Electric field intensity outside the uniformly charged solid conducting sphere
Copyright © Physics Vidyapith, Inc. All rights reserved
$\overrightarrow{dA}$
$\overrightarrow{E}$
P
r
O
R
Gaussian Surface

$$\int_{surface} \vec{E} \cdot \overrightarrow{dA} = \frac{Q_{encl}}{\epsilon_o}$$

$$\int_{surface} \vec{E} \cdot \overrightarrow{dA} = \int_{volume} (\vec{\nabla} \cdot \vec{E})\, dV$$

Cartesian $\vec{\nabla} \cdot \vec{E}\,(x,\ y,\ z) = \frac{\partial}{\partial x}E_x + \frac{\partial}{\partial y}E_y + \frac{\partial}{\partial z}E_z$

Spherical $\vec{\nabla} \cdot \vec{E}\,(r, \cancel{\theta}, \cancel{\phi}) = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2 E_r)$

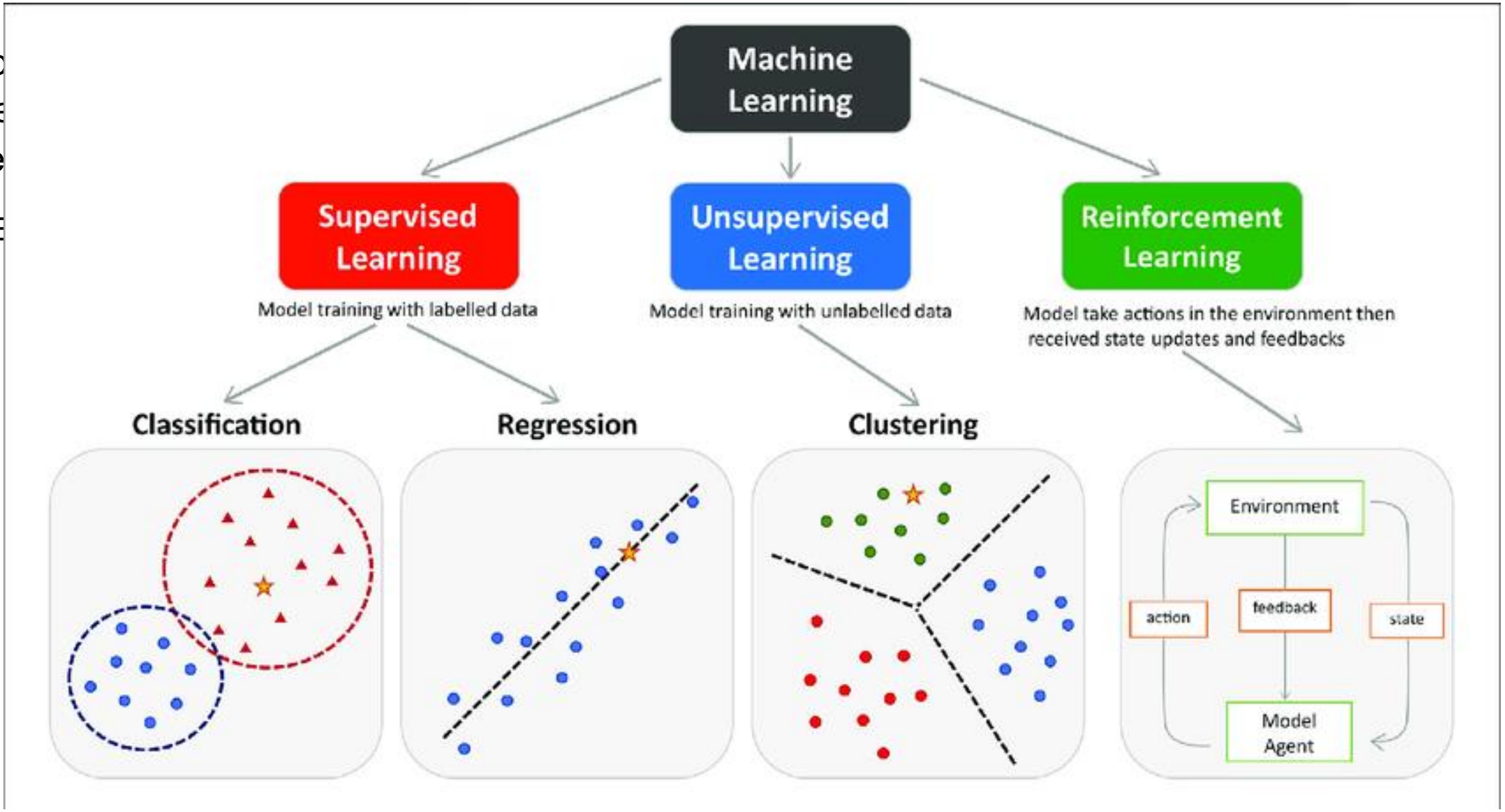
Types of Machine Learning:

Types of Machine Learning:

- Supervised learning: The machine learns from labeled data, which means the input and output are already mapped.
  - Examples: Classification, Regression, etc,

Types of Machine Learning:

- Sup
  labe
  alre
  - E

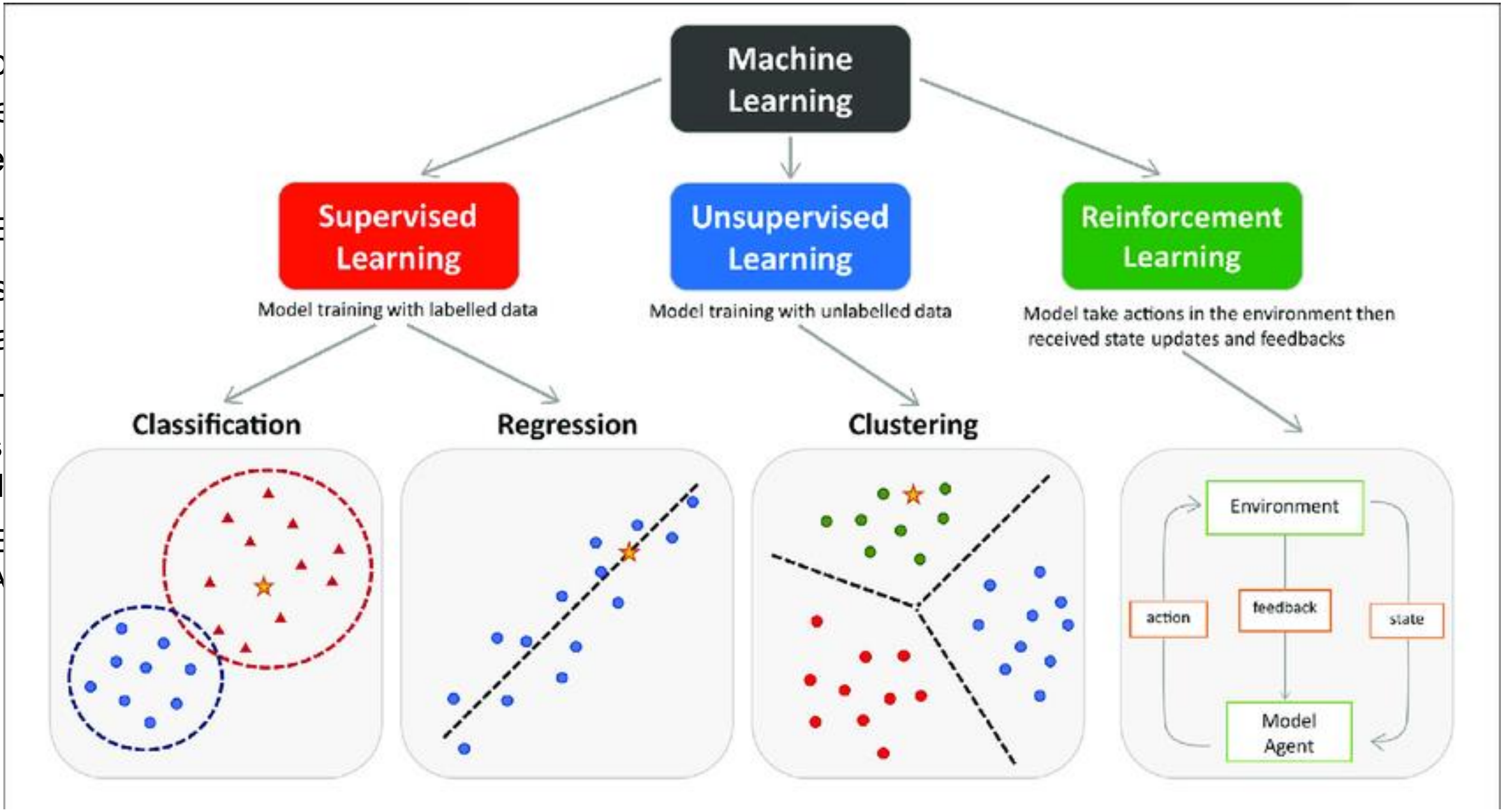


Types of Machine Learning:

- Supervised learning: The machine learns from labeled data, which means the input and output are already mapped.
  - Examples: Classification, Regression, etc,
- Unsupervised learning: The machine learns from unlabeled data, which means the output is unknown
  - The machine tries to find the hidden patterns and structures in the data, and then groups the data into different clusters
  - Examples: Clustering, Dimensionality Reduction, Anomaly Detection, etc.

Types of Machine Learning:

- Sup
  labe
  alre
  - E
- Uns
  unla
  - T
    s
    d
  - E
    A

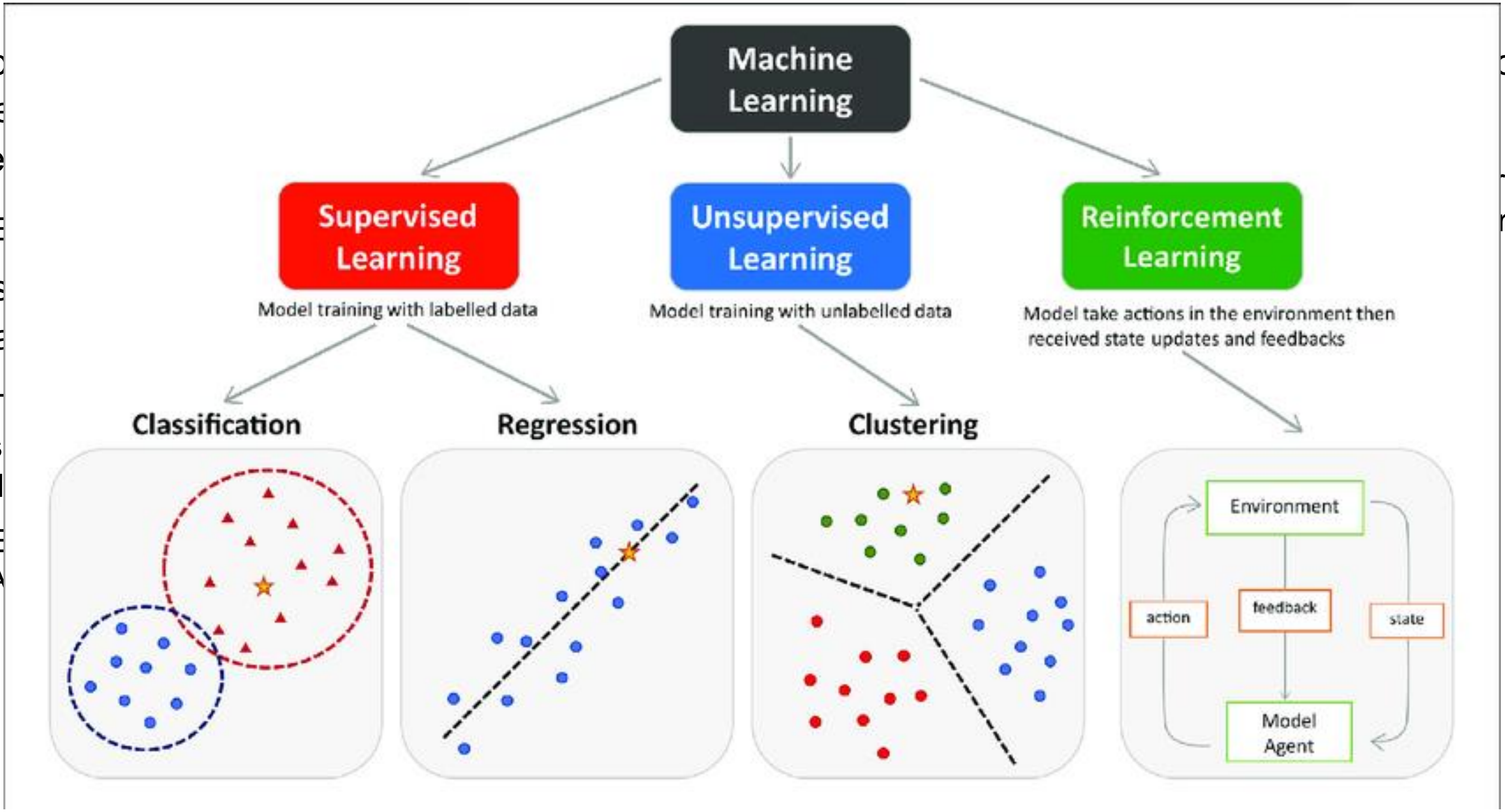


Types of Machine Learning:

- Supervised learning: The machine learns from labeled data, which means the input and output are already mapped.
  - Examples: Classification, Regression, etc,
- Unsupervised learning: The machine learns from unlabeled data, which means the output is unknown
  - The machine tries to find the hidden patterns and structures in the data, and then groups the data into different clusters
  - Examples: Clustering, Dimensionality Reduction, Anomaly Detection, etc.
- Semi-supervised learning: Only some of the input and output are mapped.
  - Uses the labelled data to learn the features of the data and then uses the unlabeled data to improve the learning
  - . Examples: Image Recognition, Speech Recognition, Sentiment Analysis, etc.

Types of Machine Learning:

- Sup of the input labe alre res of the data rove the
  - E
- Uns unla Recognition,
  - T s d
  - E A



Types of Machine Learning:

- Supervised learning: The machine learns from labeled data, which means the input and output are already mapped.
  - Examples: Classification, Regression, etc,
- Unsupervised learning: The machine learns from unlabeled data, which means the output is unknown
  - The machine tries to find the hidden patterns and structures in the data, and then groups the data into different clusters
  - Examples: Clustering, Dimensionality Reduction, Anomaly Detection, etc.
- Semi-supervised learning: Only some of the input and output are mapped.
  - Uses the labelled data to learn the features of the data and then uses the unlabeled data to improve the learning
  - . Examples: Image Recognition, Speech Recognition, Sentiment Analysis, etc.
- Reinforcement learning: The machine learns from its own actions and feedback, which means there is no fixed data set
  - The machine tries to explore different actions and outcomes in an environment and then learns from the rewards or penalties it receives. Examples: Self-driving Cars, Game Playing, Robotics, etc.

Types of Machine Learning:

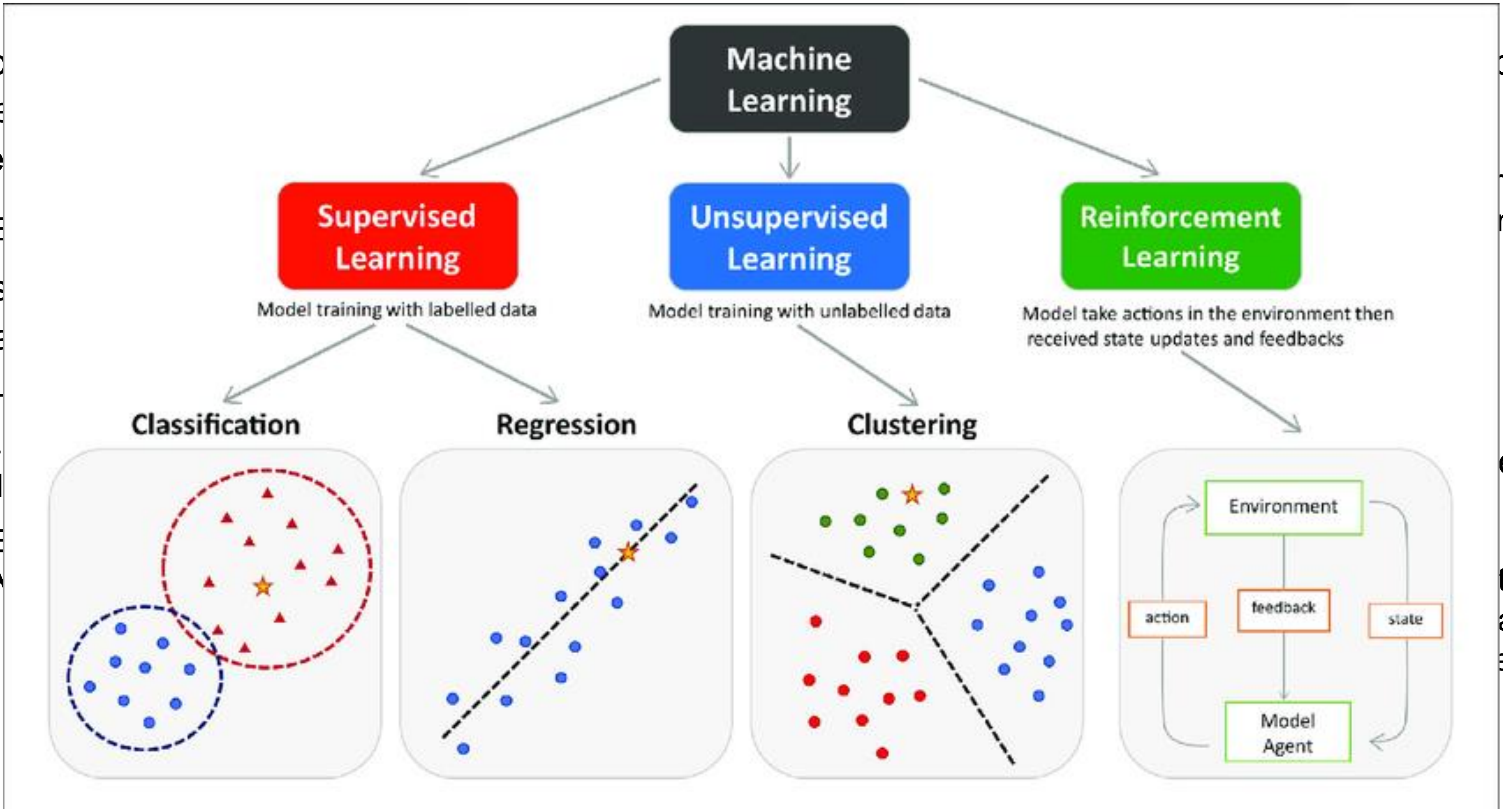
- Sup of the input labe alre
  - E
- Uns unla
  - T s d
  - E A

res of the data rove the

Recognition,

earns from eans there is

tions and arns from the es: Self-driving

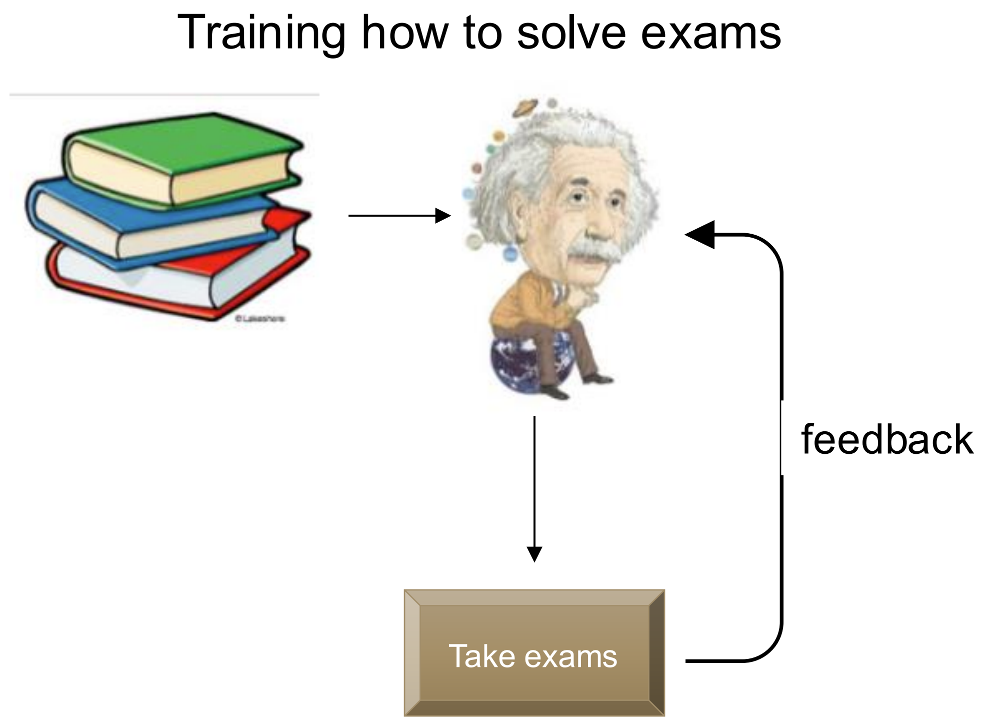


ML workflow

Machine Learning Workflow

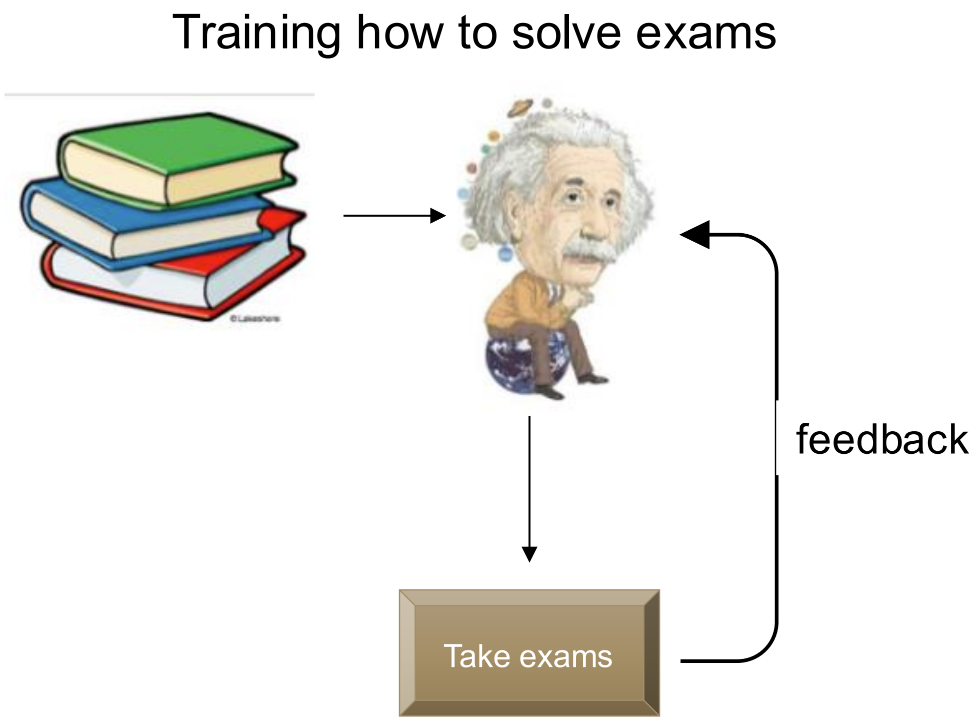
ML workflow

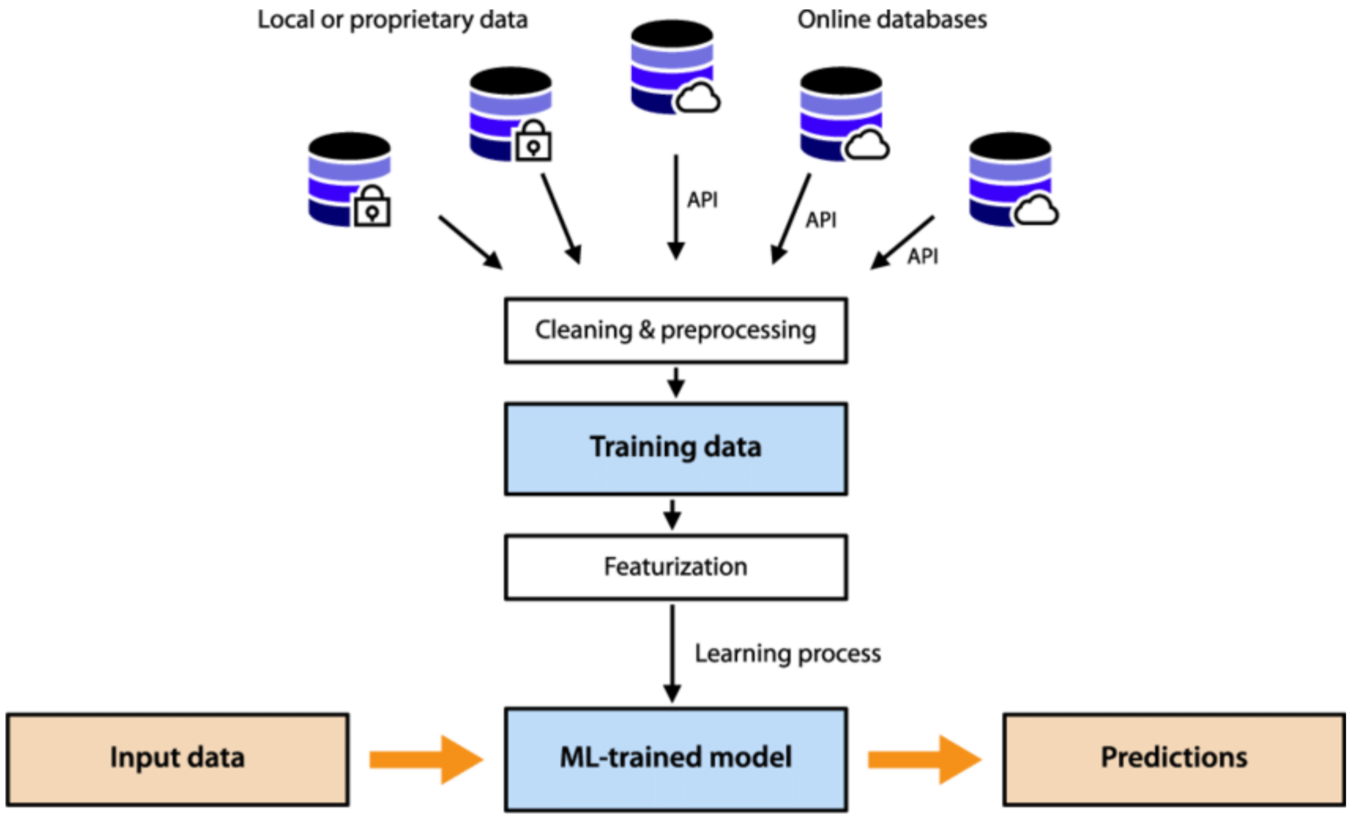
Machine Learning Workflow



ML workflow

Machine Learning Workflow

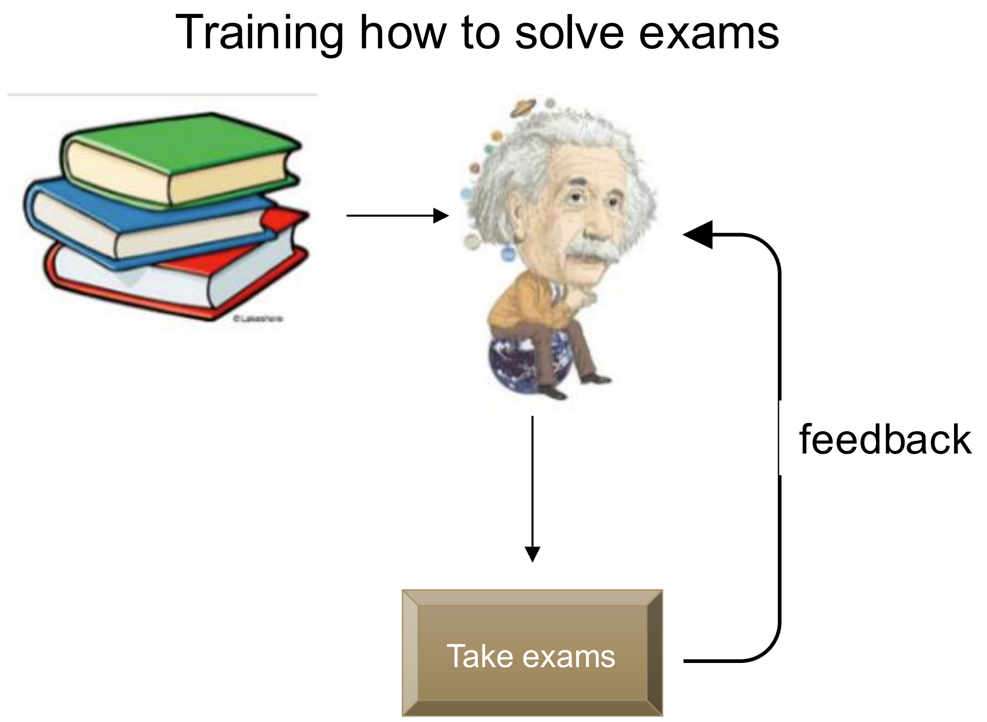


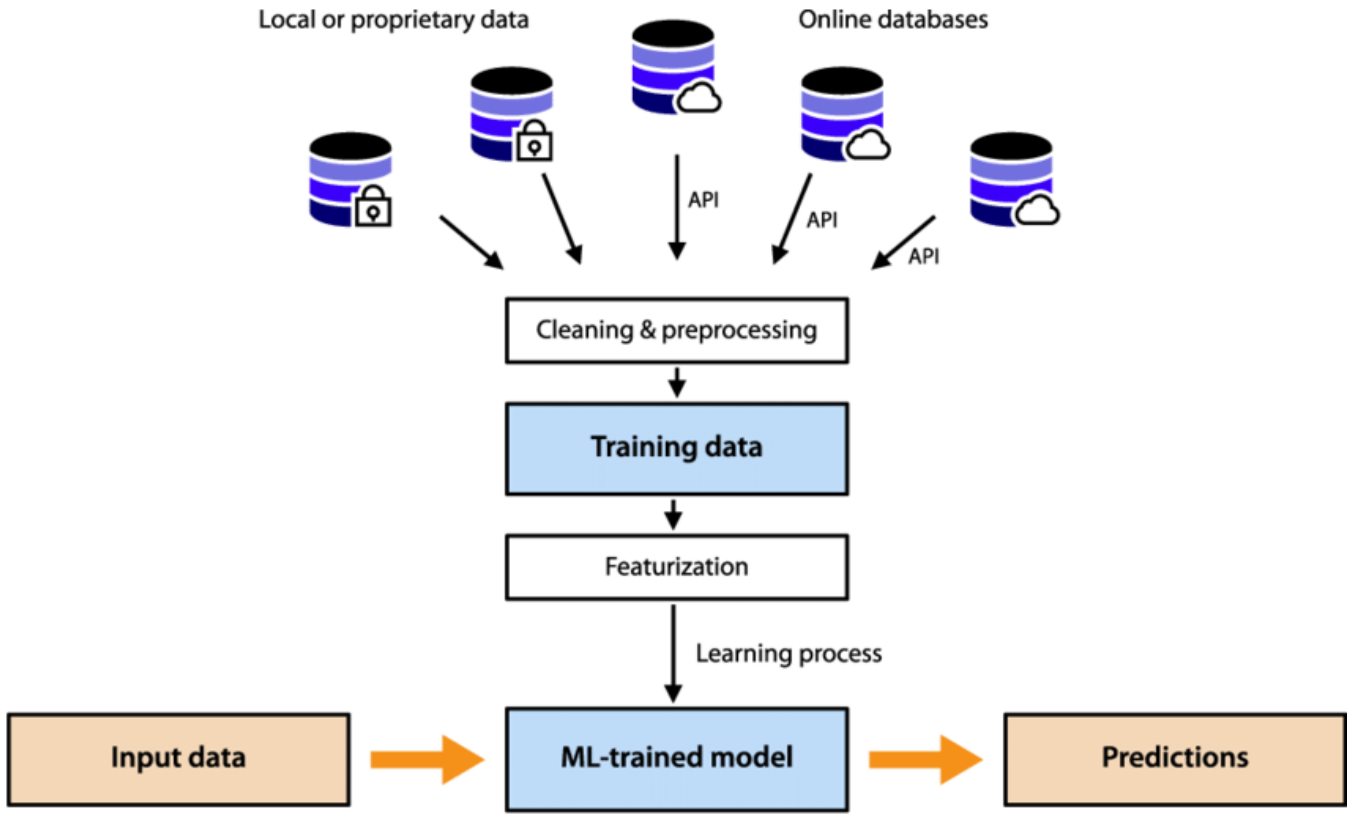


ML workflow

Machine Learning Workflow

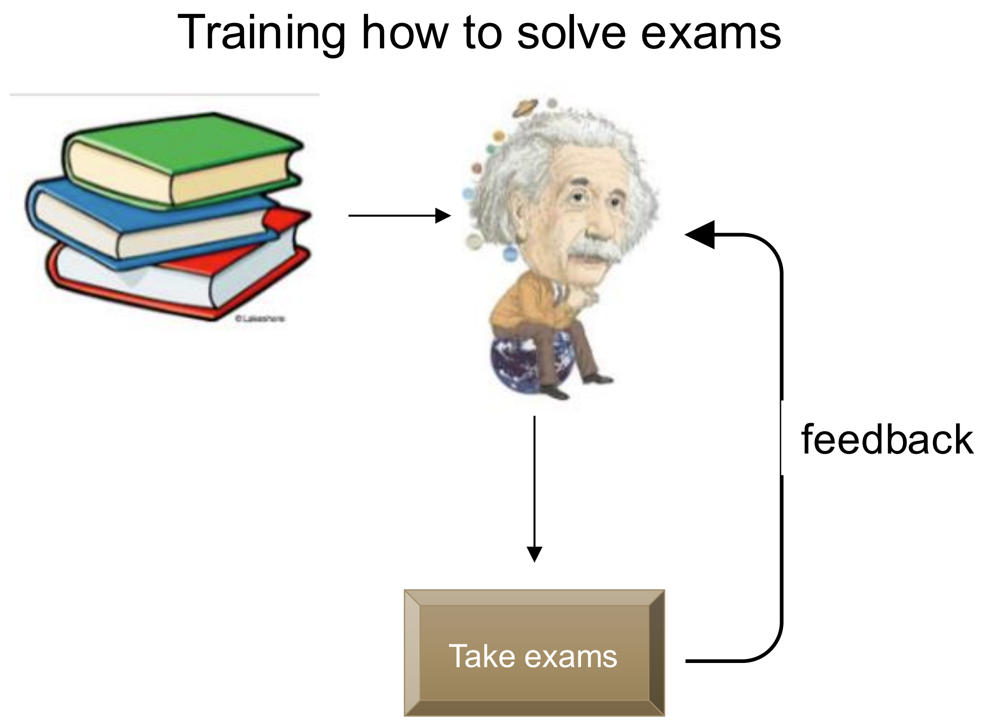
1. Data Collection and Preprocessing

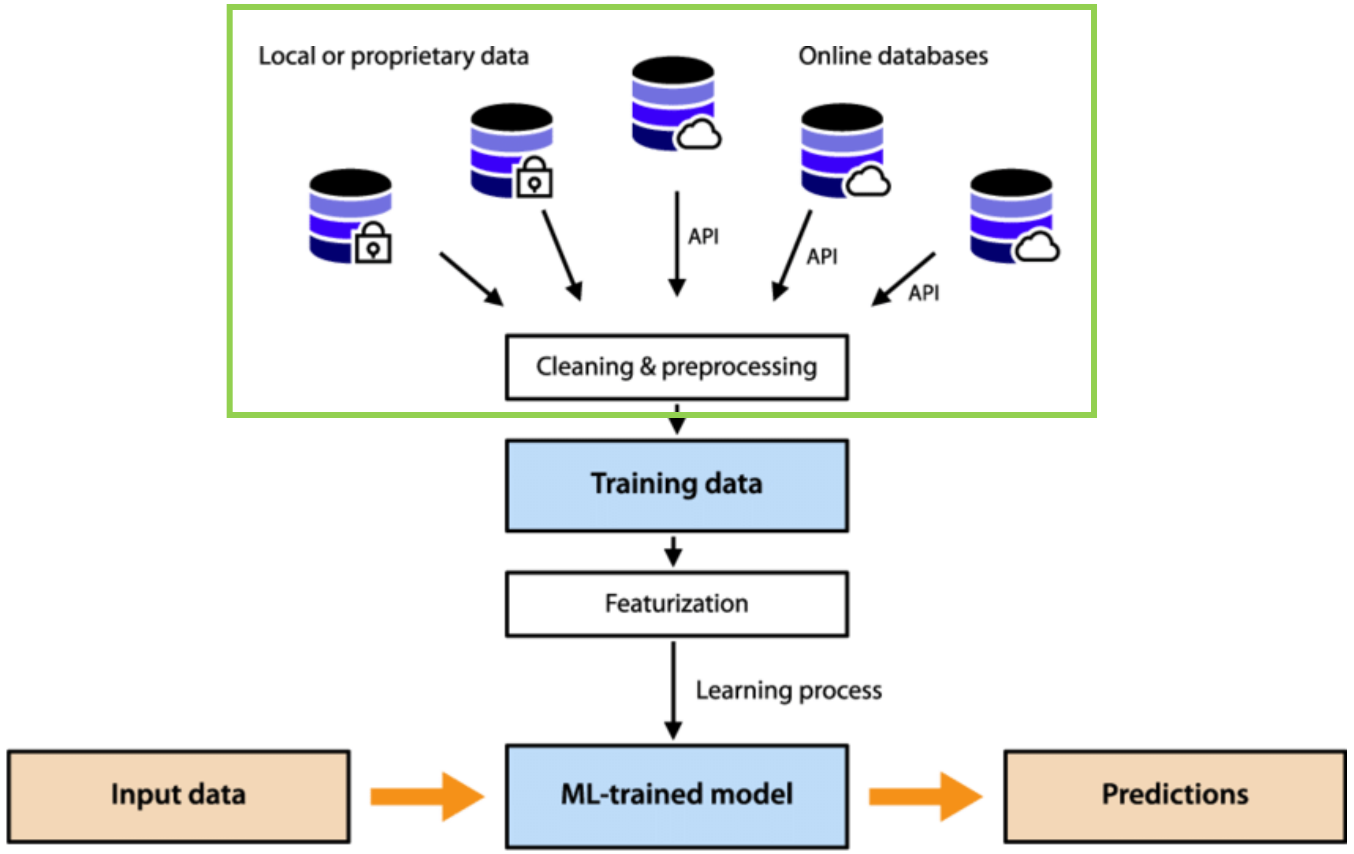




ML workflow

Machine Learning Workflow

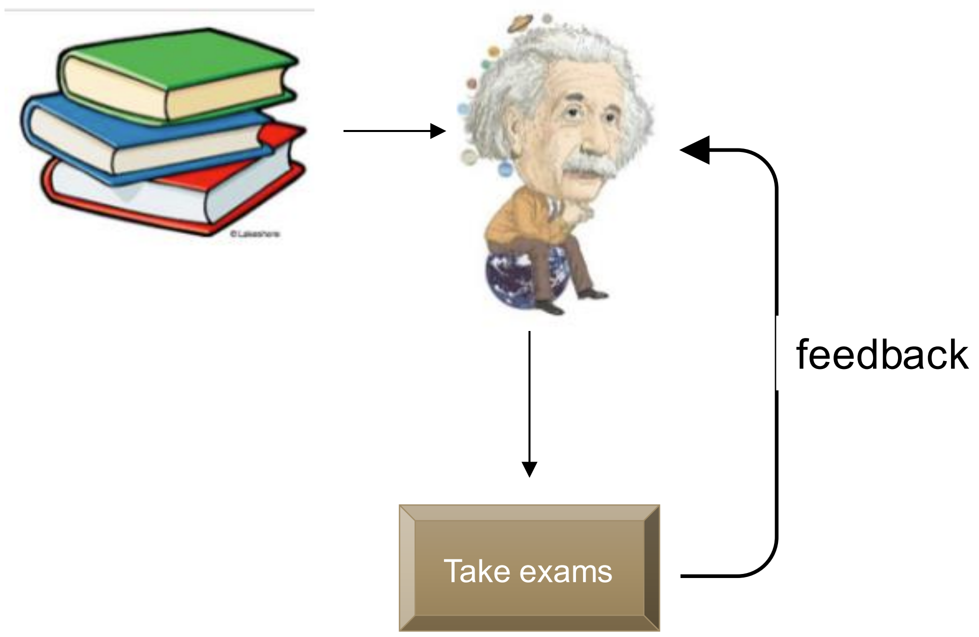


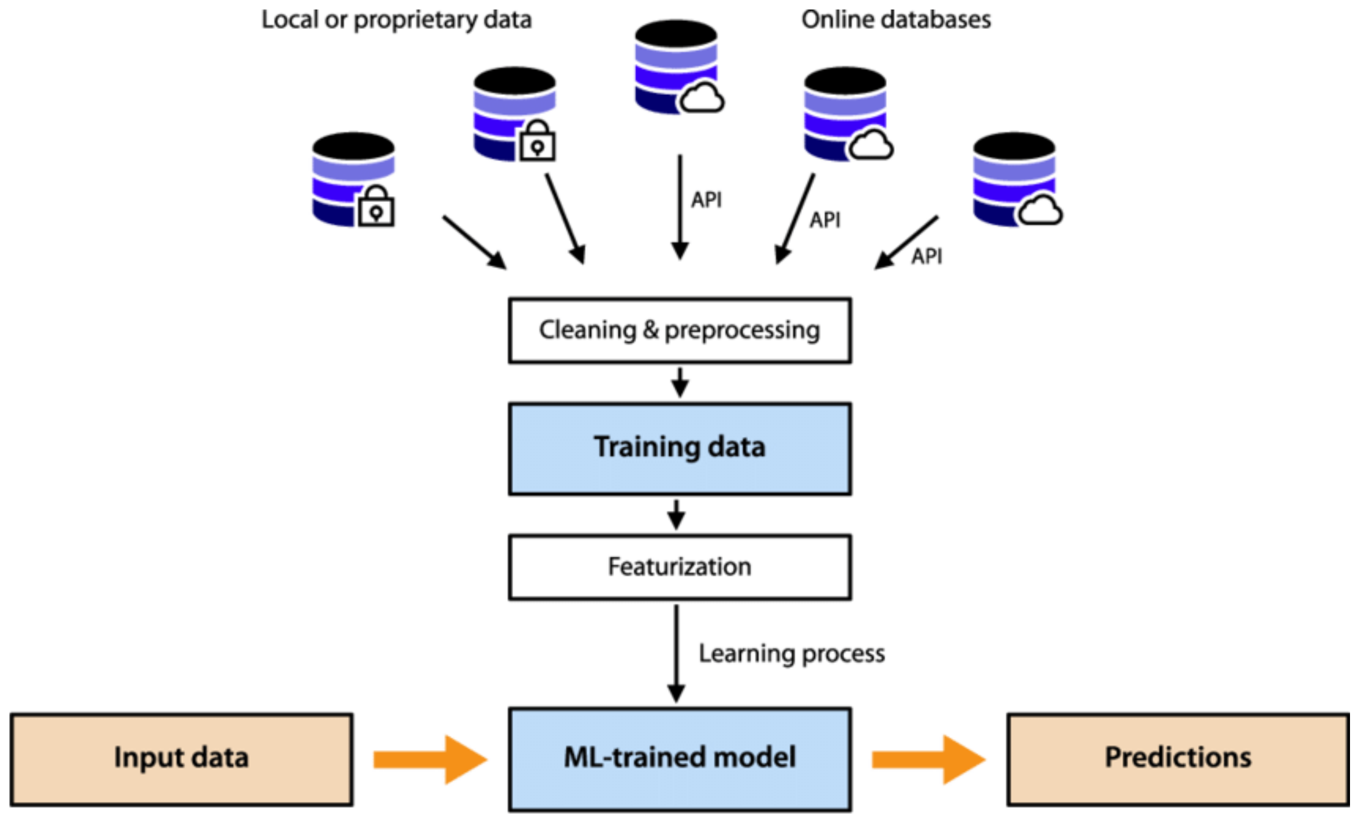


ML workflow

Machine Learning Workflow

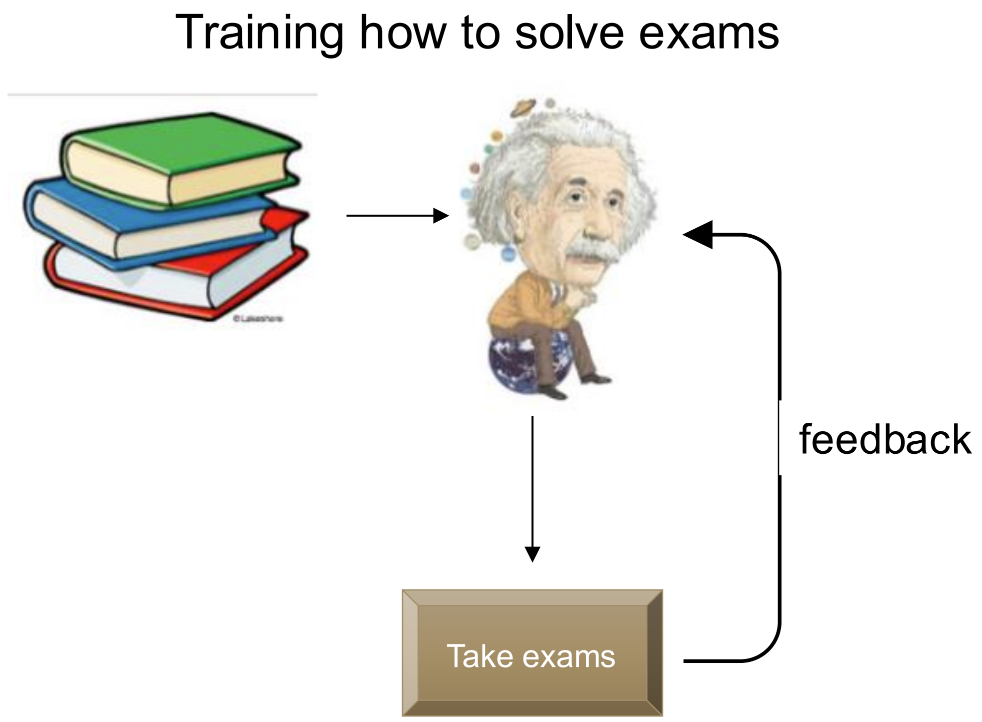
Training how to solve exams

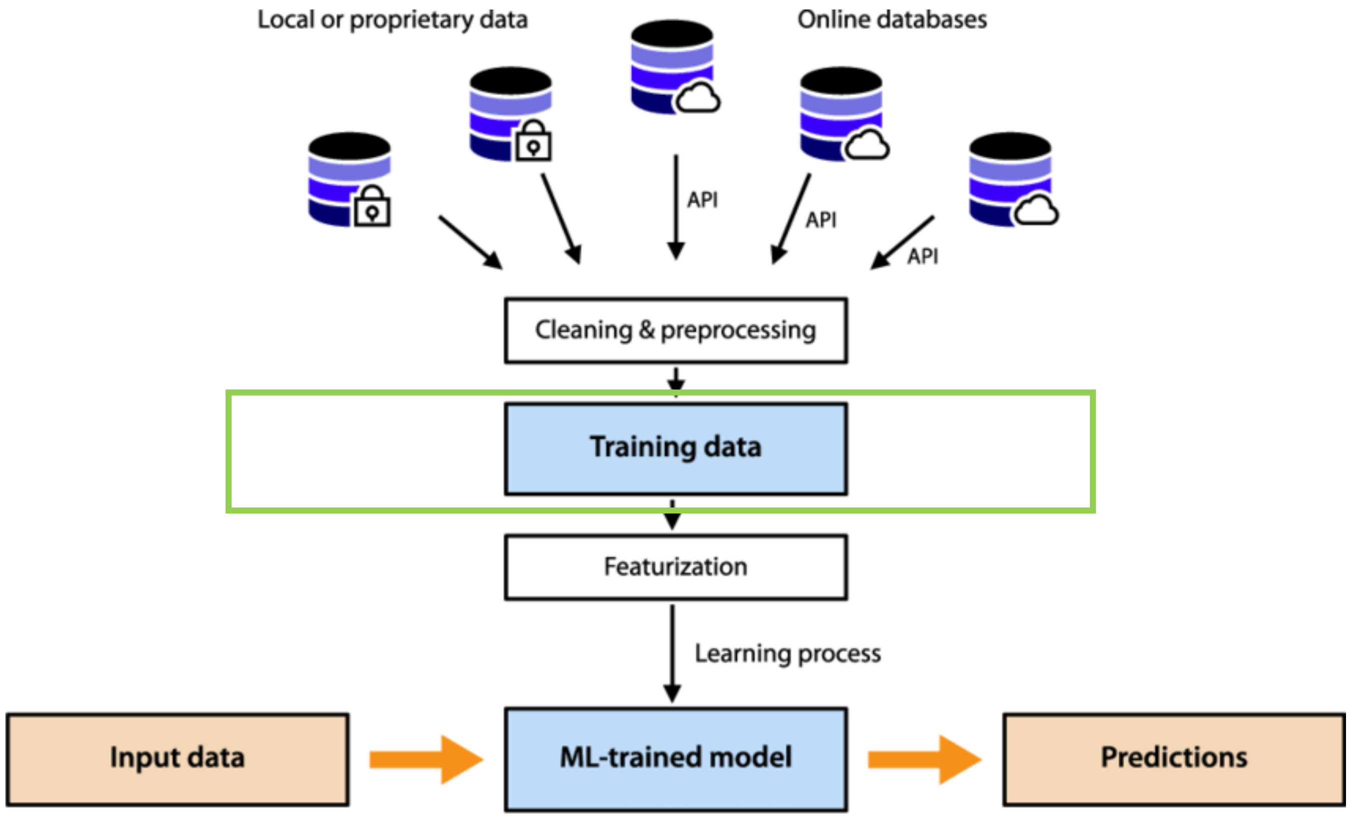




ML workflow

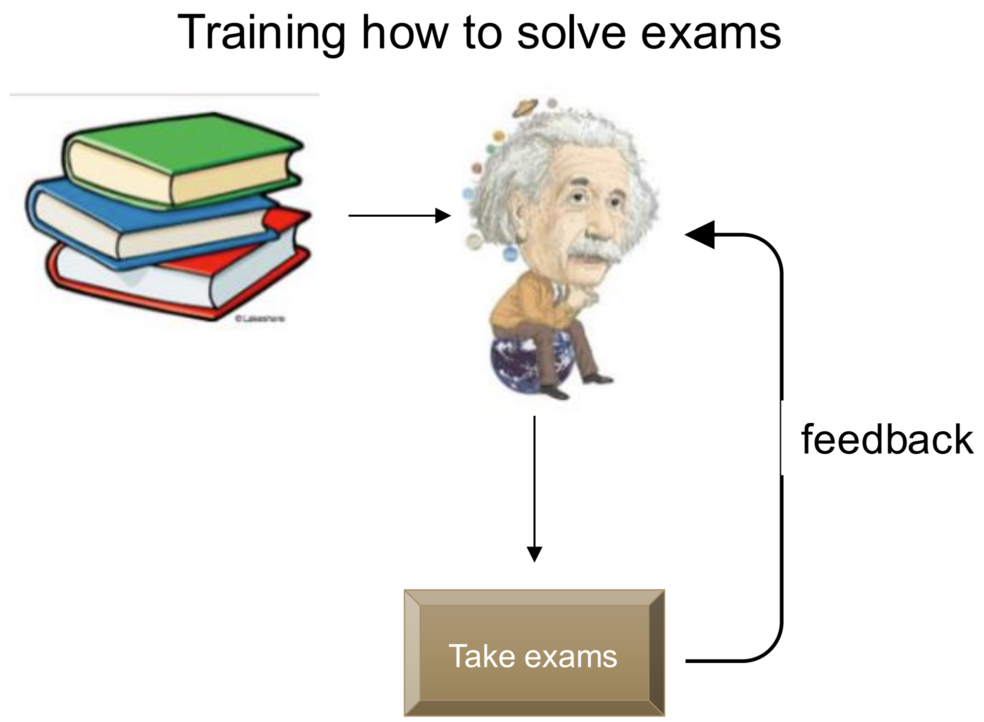
Machine Learning Workflow

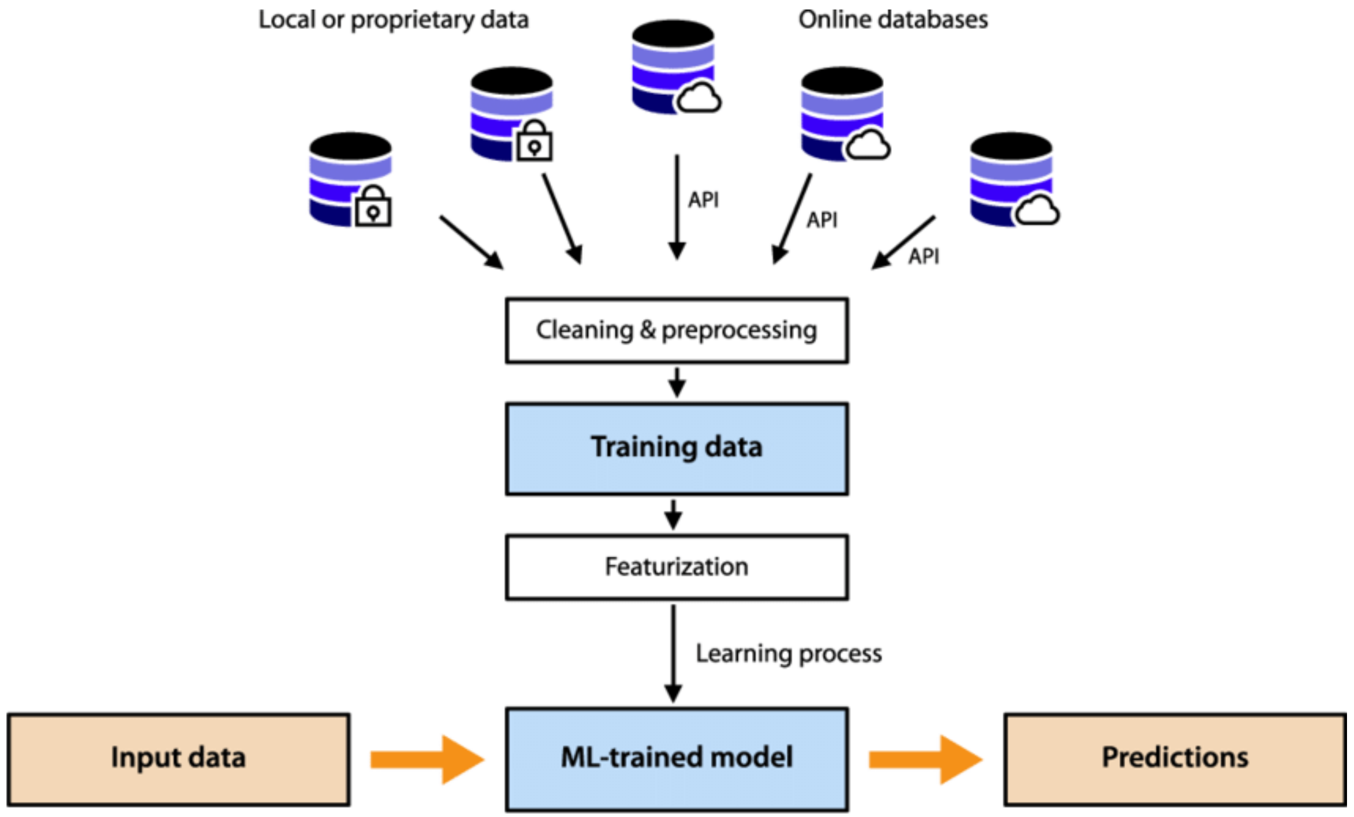




ML workflow

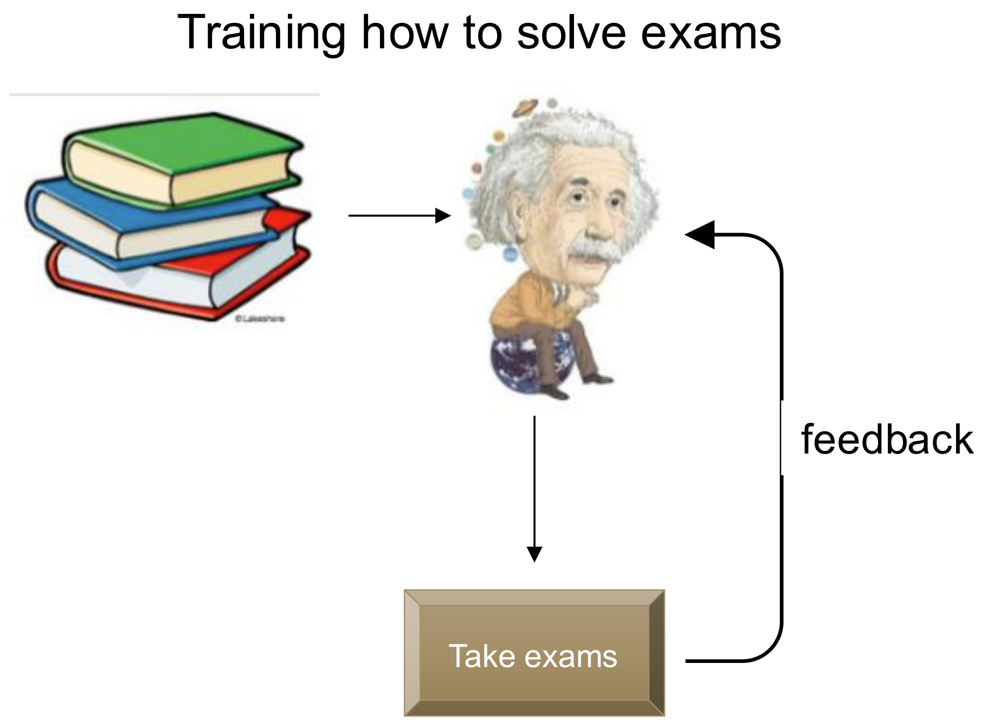
Machine Learning Workflow

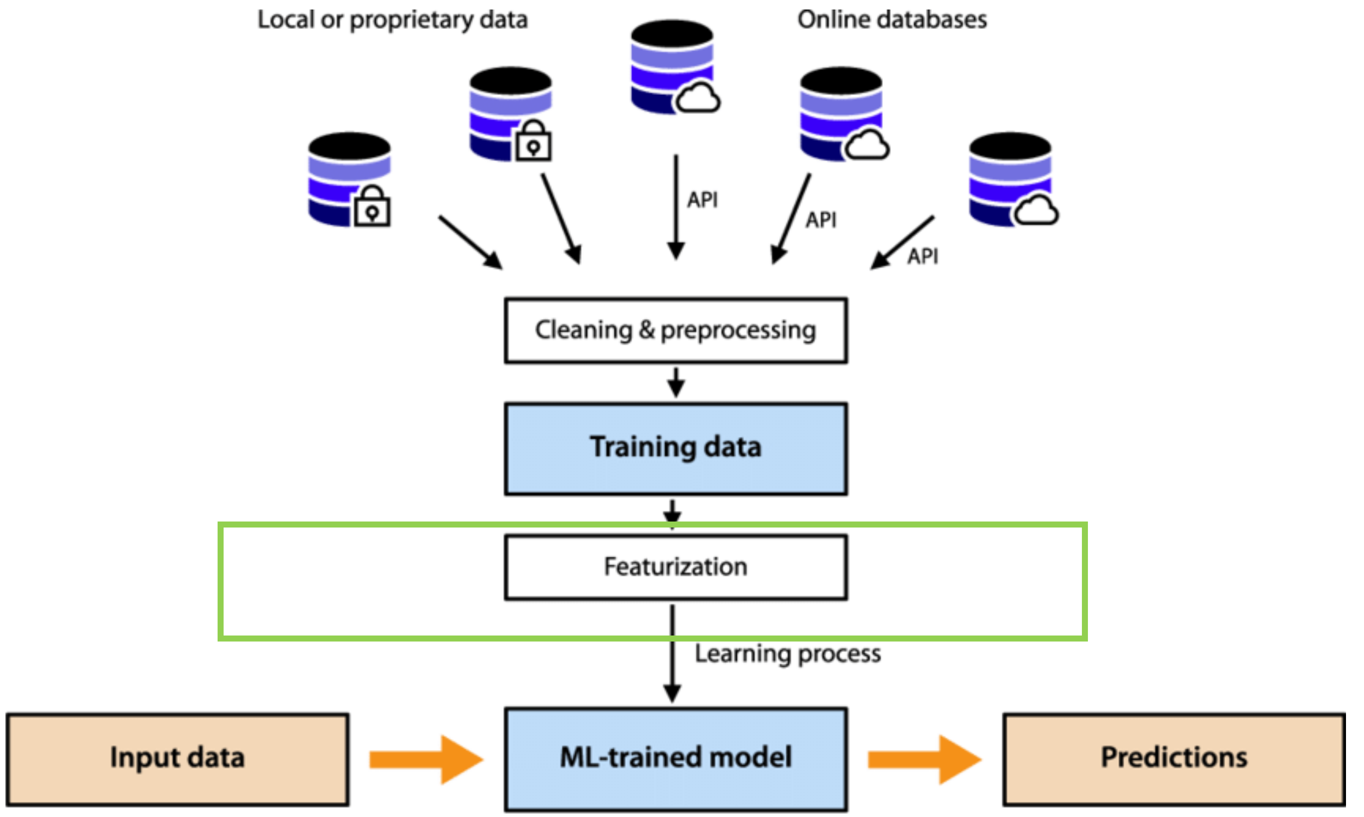




ML workflow

Machine Learning Workflow

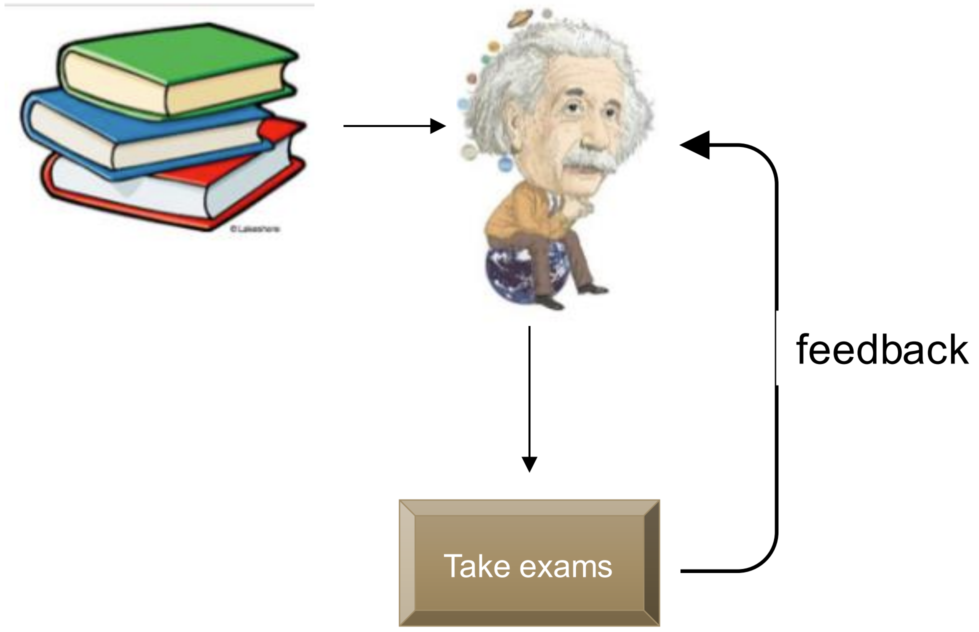


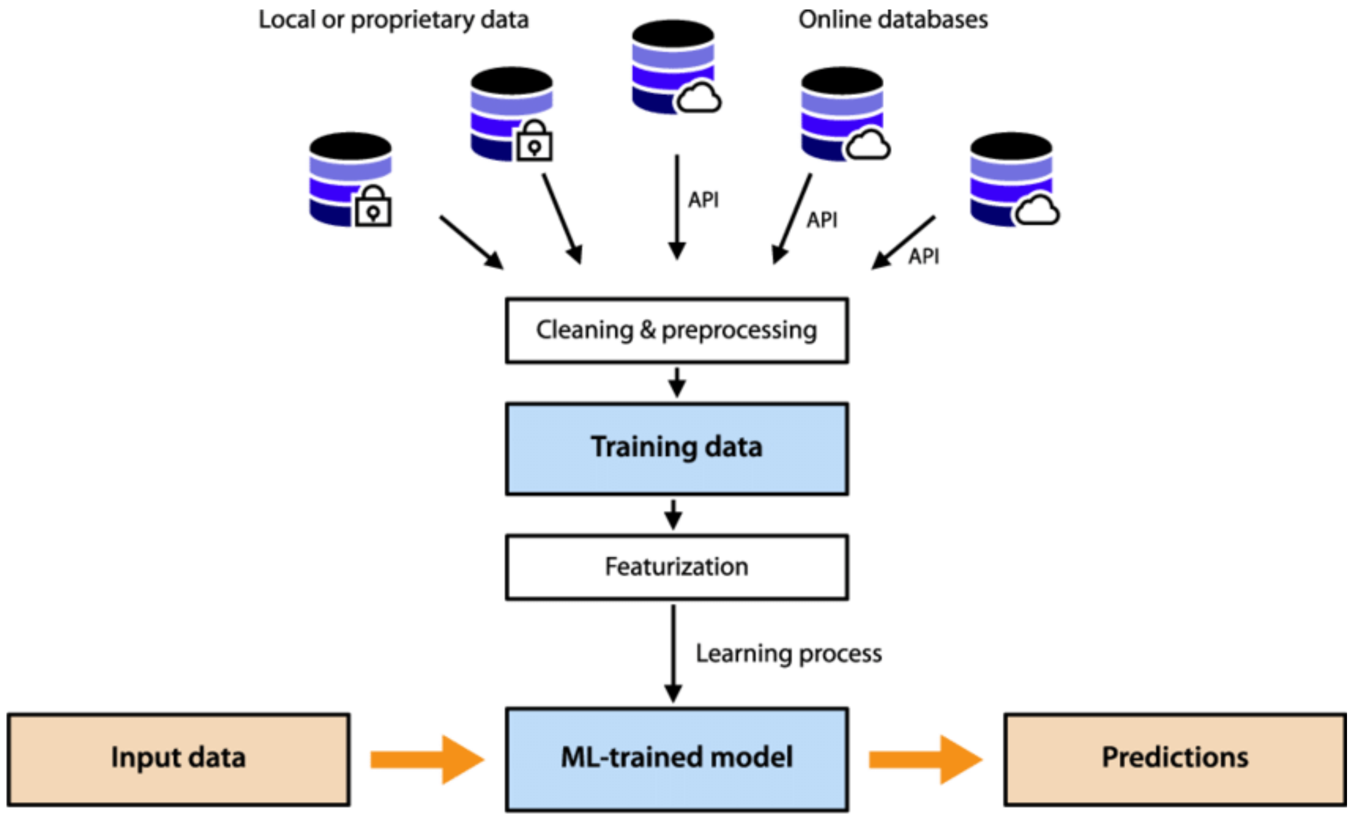


ML workflow

Machine Learning Workflow

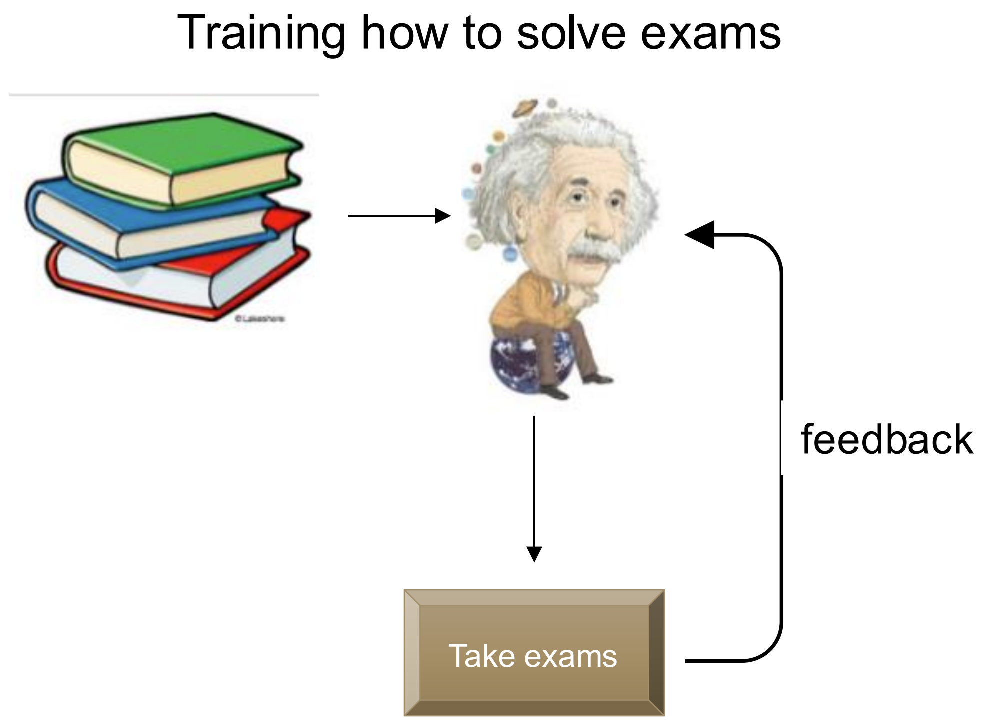
Training how to solve exams

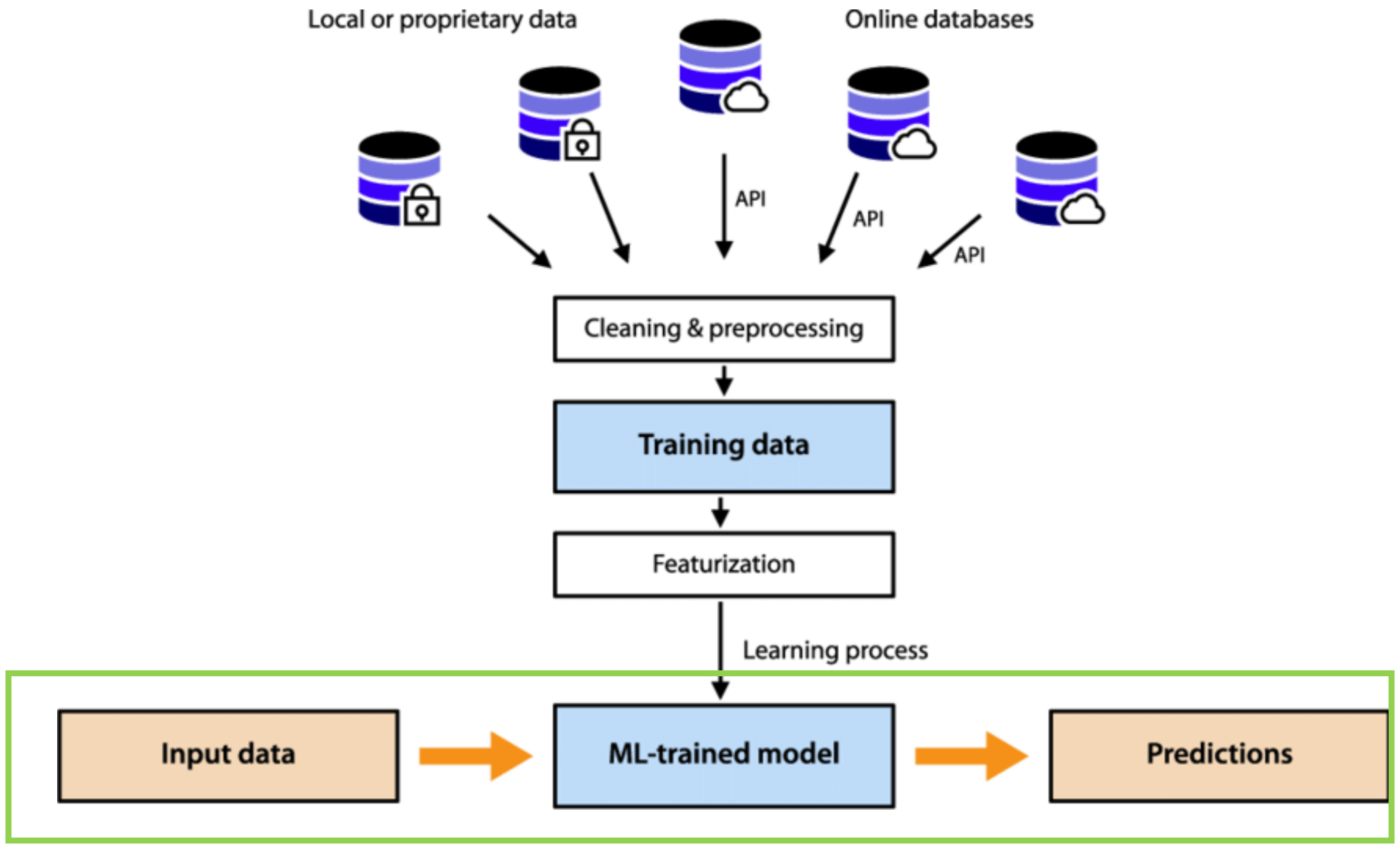




ML workflow

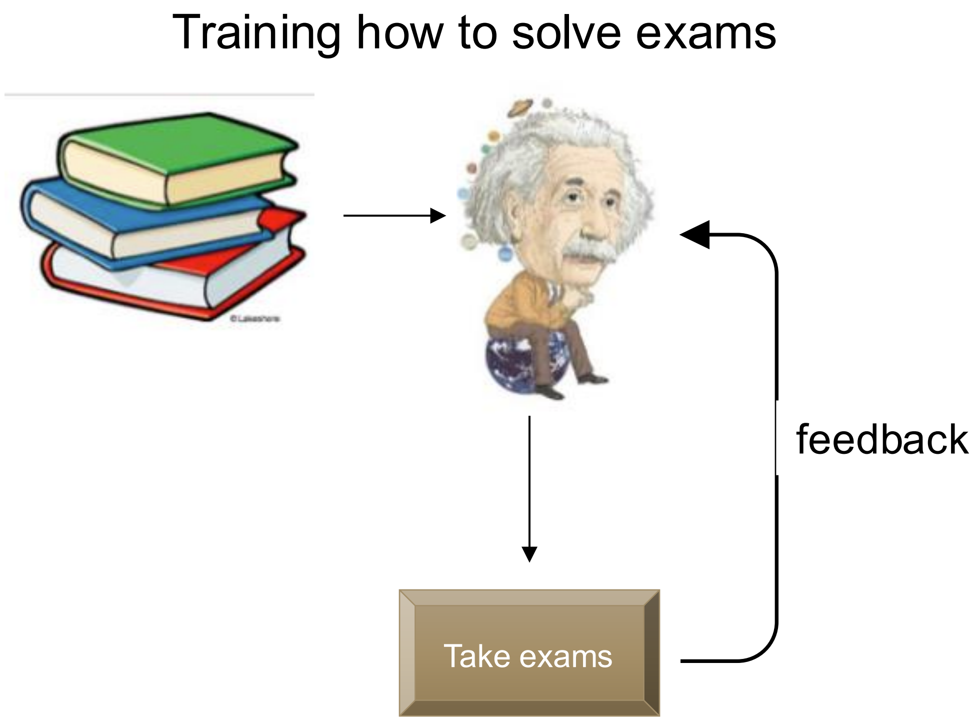
Machine Learning Workflow

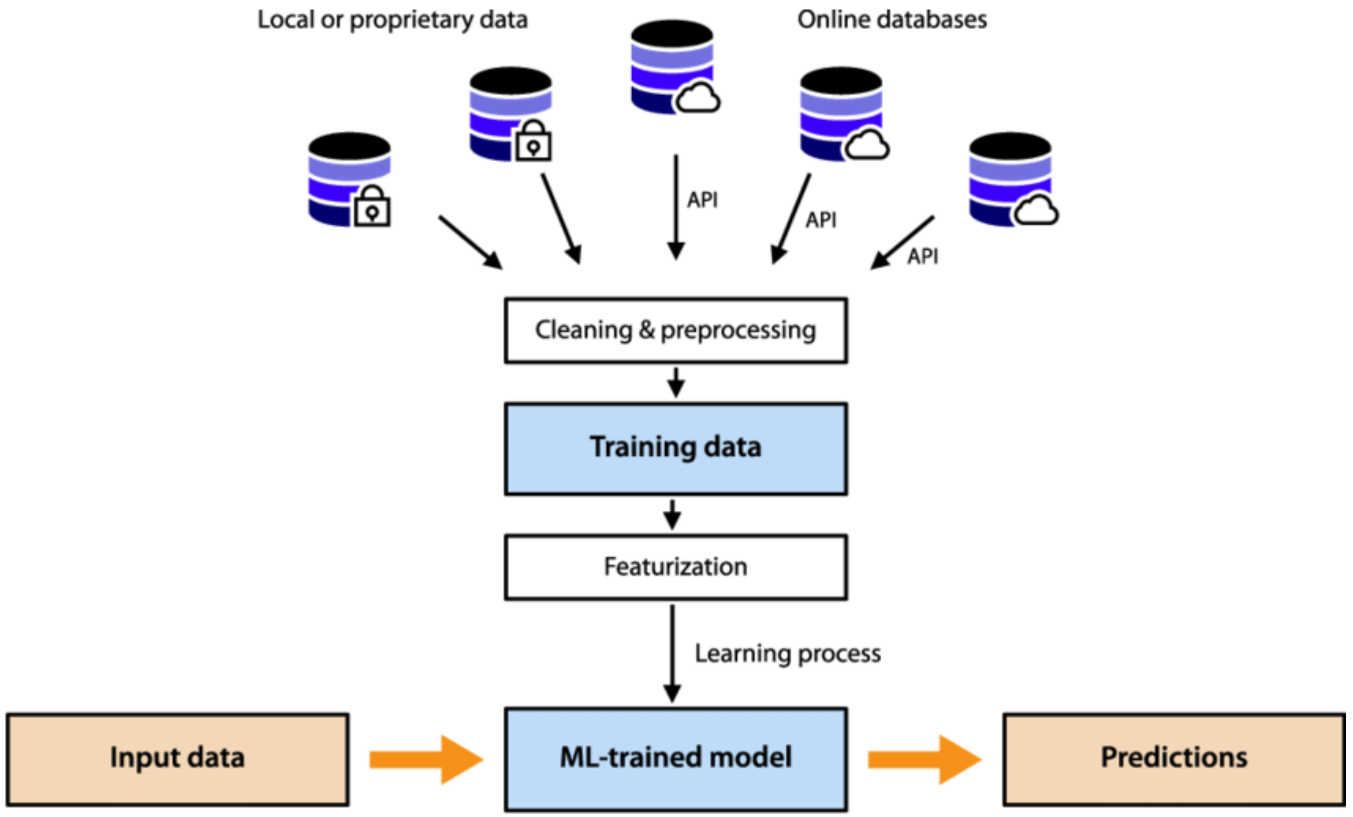




ML workflow

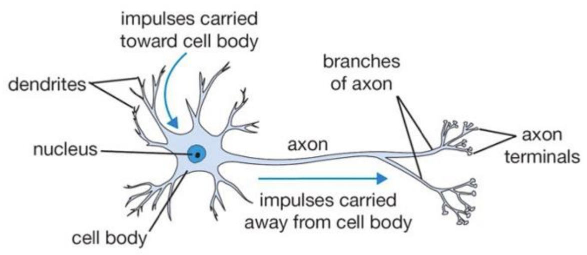
Machine Learning Workflow

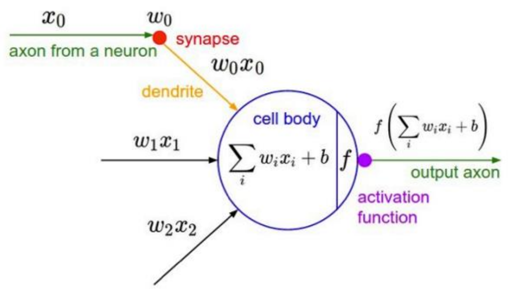




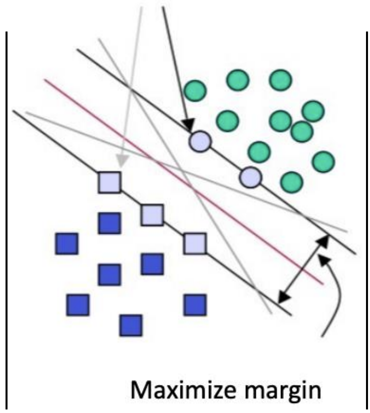
Machine learning is not just Neural

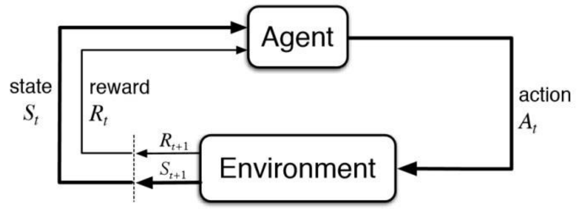
Support vector machine SVM


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

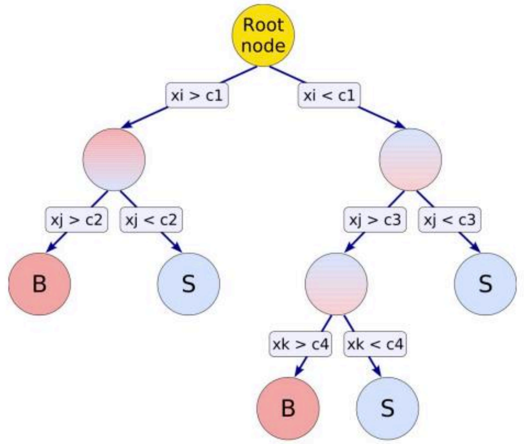


Neural Networks

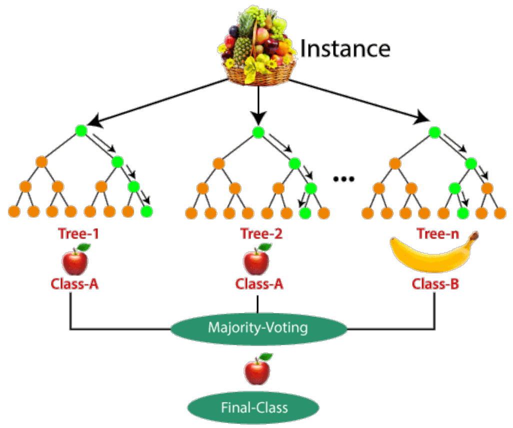




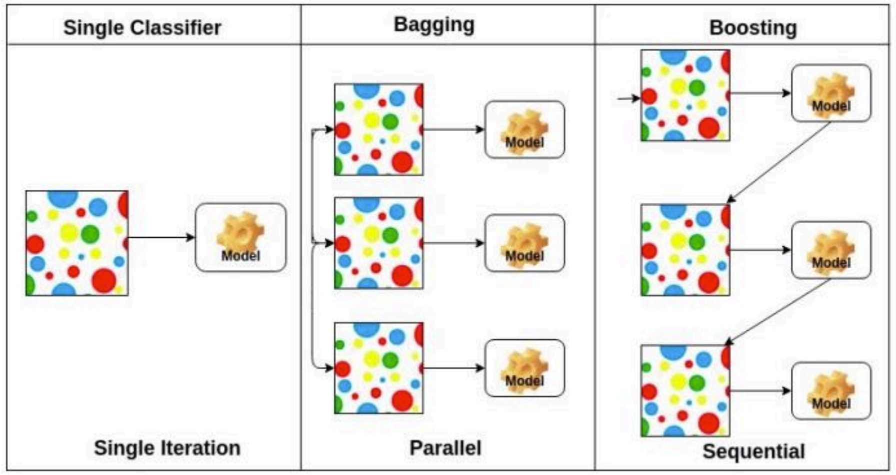
Reinforcement learning



Decision Tree



Random forest
Bootstrapped samples



Bagging and Boosting

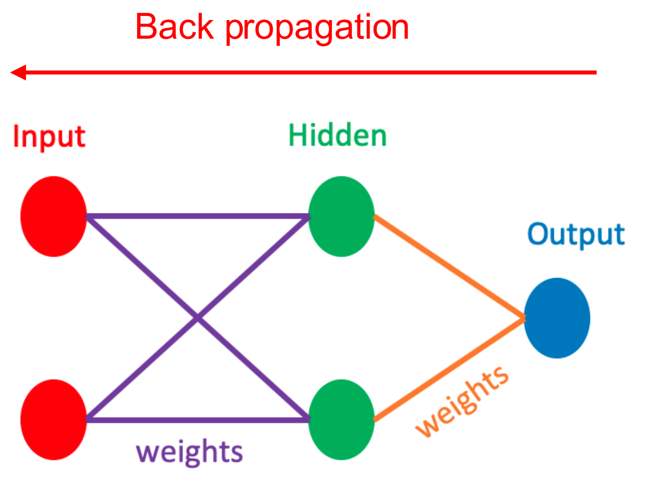
Easy to build, easy to use and easy to interpret

Artificial Neural Networks (ANNs)

- Deep neural networks
  - Neural network with more than 2 layers
  - Can model more complex functions

- Let's start with a simple example:
  - X, W_s, b_s can be numbers, vectors, matrix, or tensor
  - f_{NN} is the output calculated (Forward Propagation)
  - What if the output deviates from the desired value?

- Important: A neural network is only some mathematical function, no magic behind
- Training : finding the best parameters for a given task



$$f_{NN} = \sigma (b_2 + W_2 \sigma (b_1 + W_1 X))$$

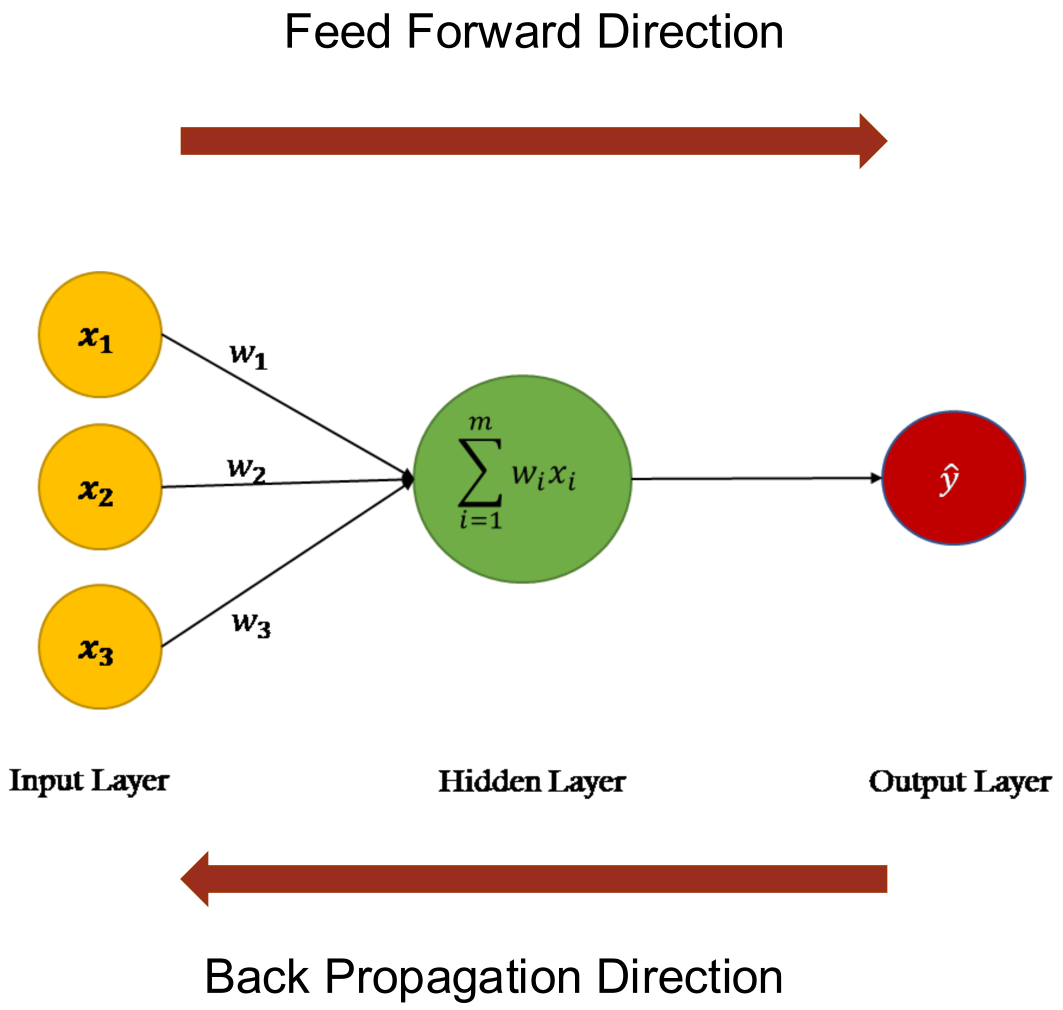
Feed forward

Theory of ANN

- An artificial neural network in a supervised learning algorithm which means that we provide it the input data containing the independent variables and the output data that contains the dependent variable.
- For instance, in our example our independent variables are **X** = [x_1, x_2, x_3]
- The dependent variable is Y.
- Thus **X** -> Y, where **X** is a vector of data or features, and Y is a label
- In the beginning, the ANN makes some random predictions, these predictions are compared with the correct output and the error (the difference between the predicted values and the actual values) is calculated.

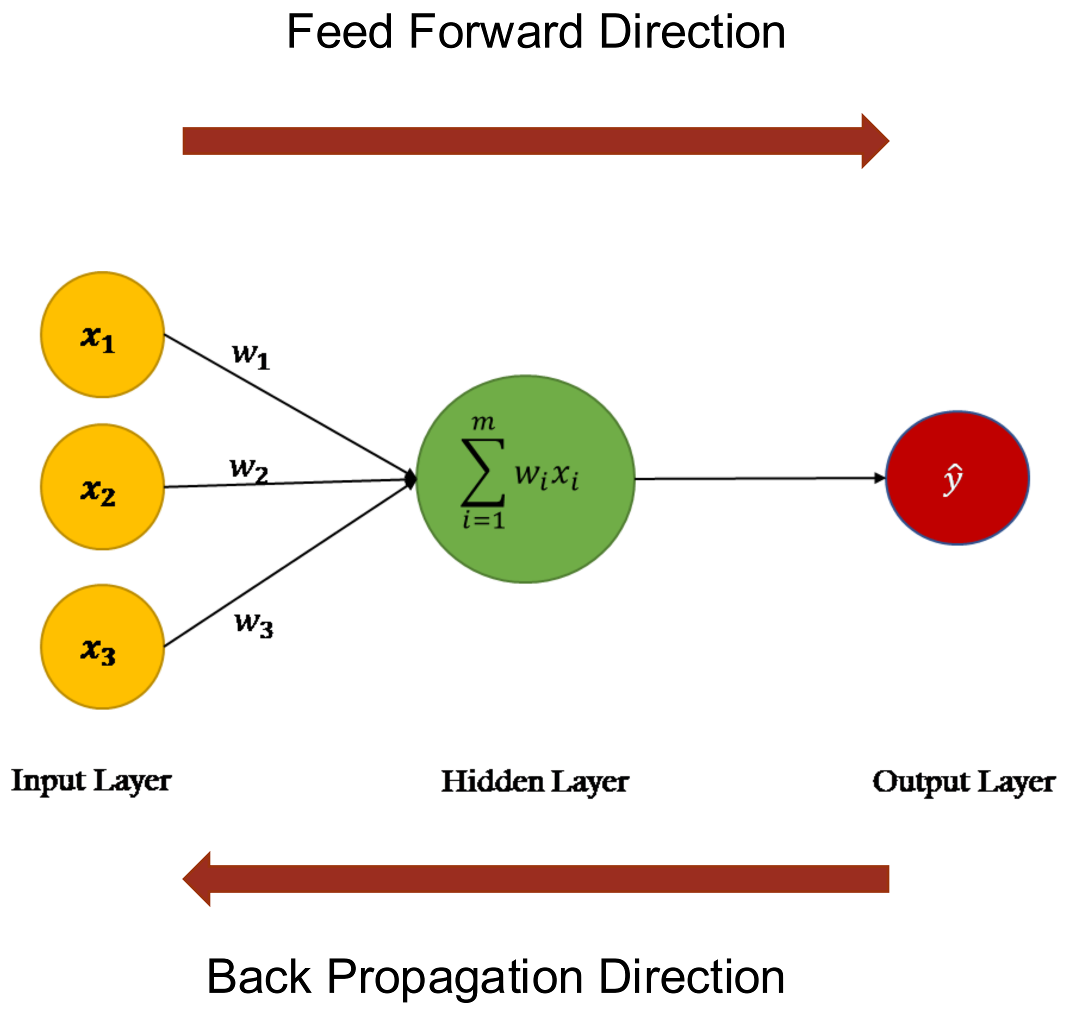
Theory of ANN

- The function that finds the difference between the actual value and the propagated values is called the cost function.
- The cost here refers to the error.
- Our objective is to minimize the cost function.
    - Training a neural network basically refers to minimizing the cost function.
- We will see how we can perform this task.
- A neural network executes in two phases: Feed Forward phase and Back Propagation phase.



Theory of ANN

- In the feed-forward phase of ANN, predictions are made based on the values in the input nodes and the weights.
- If you look at the neural network in the figure, you will see that we have three features in the dataset: x_1, x_2, and x_3, therefore we have three nodes in the first layer, also known as the input layer.
- The weights of a neural network are basically the wires that we have to adjust in order to be able to correctly predict our output.
- For now, just remember that for each input feature, we have one weight.

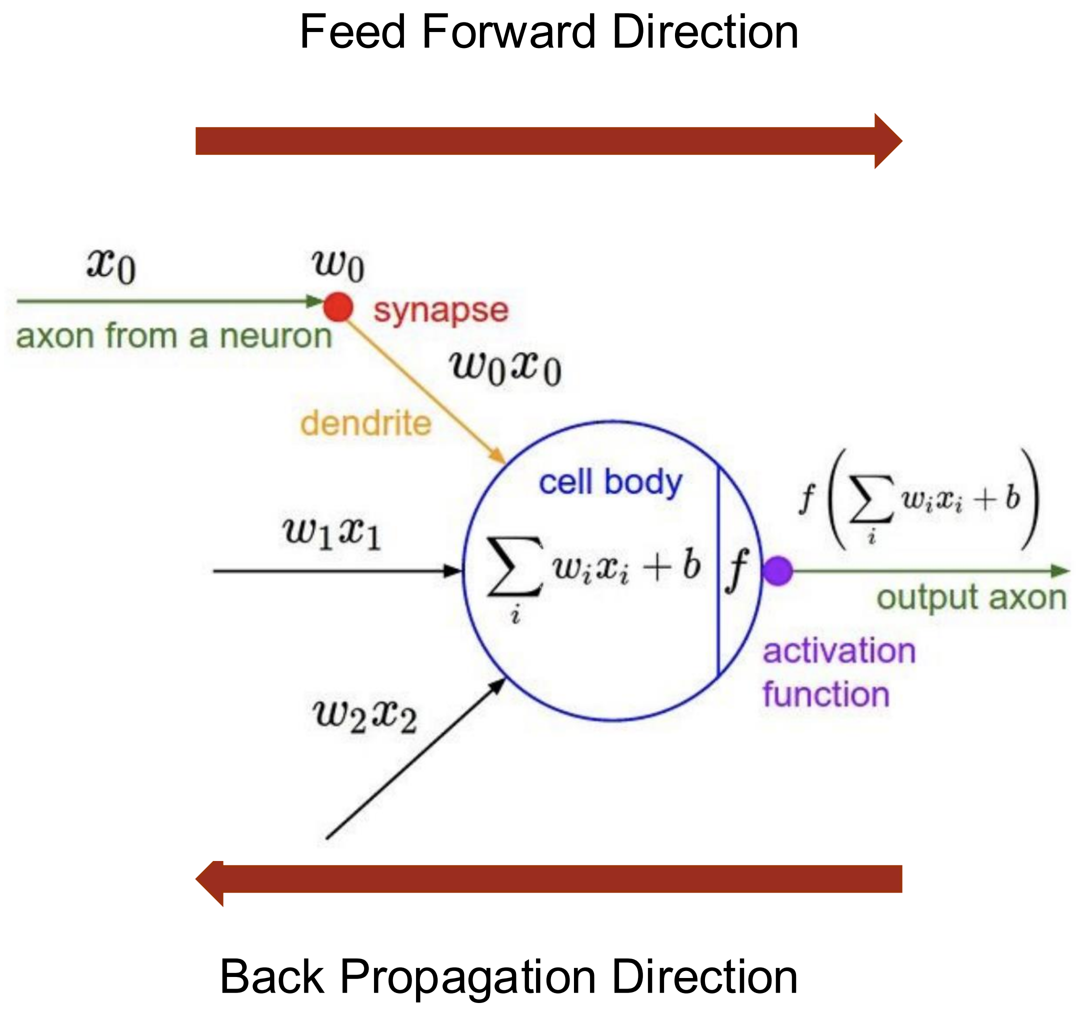


Feedforward Phase of ANN

- We start by calculating the dot product between inputs and weights
- The nodes in the **input layer** are connected with the output layer via three weights.
- In the output layer, the values in the input nodes are multiplied by the corresponding weights and are added together.
- Finally, the bias term or threshold term b is added to the sum.
- Mathematically, the summation of the dot product is:

  X*W=$x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + b$

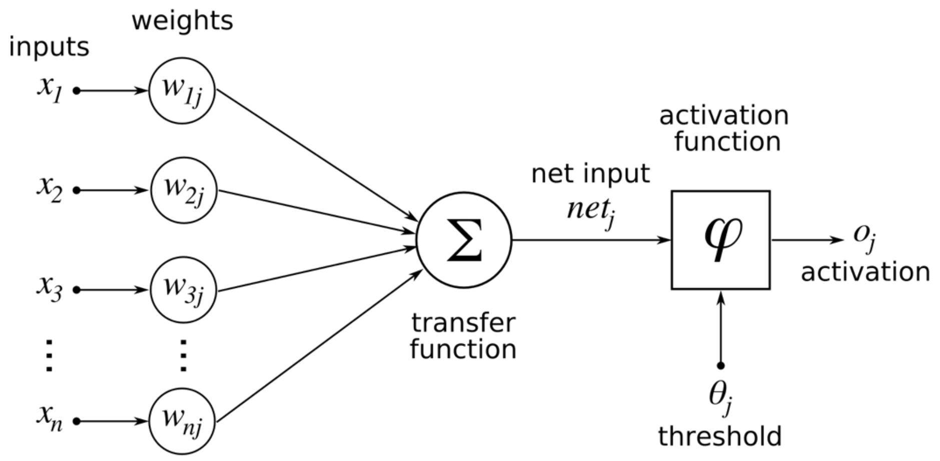
- Note that if there were no bias term, the neuron would fire even if there was no input, which is unphysical (or un-biological)

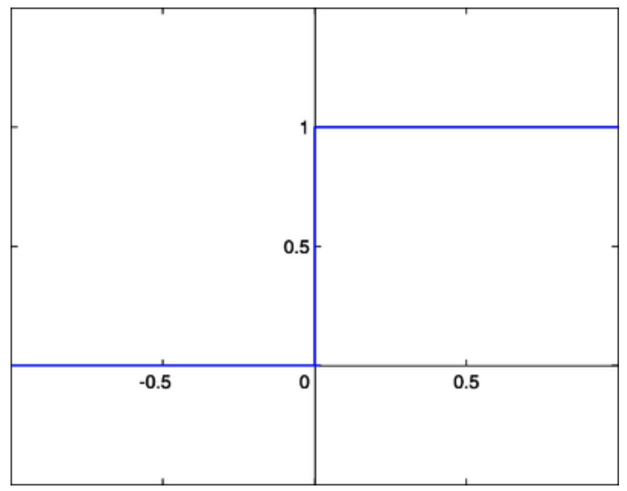


Activation Term

- Next, we pass the summation of dot products (X.W) plus bias term b through an activation function
- The dot product **X*W** + b can produce any set of values.
- However, in our output, we want to have values in a restricted range, typically between 0 and 1.
- To accomplish this, we need an activation function, which restricts the input values to lie between 0 and 1.
- Most common activation function is the sigmoid.
- This restriction has lately been relaxed with the reLu activation function, for example

Activation Function: Unit Step

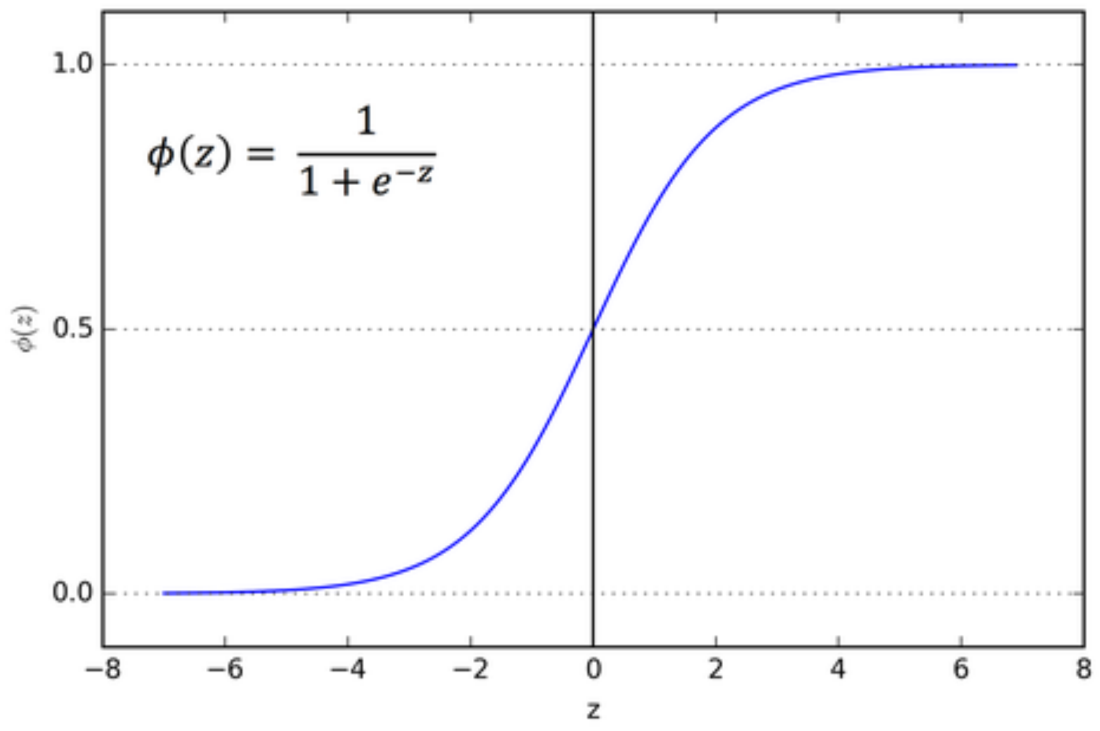




- The unit step function returns 0.0 when the input is less than or equal to 0.
- It returns a value 1 if the input is greater than 0.

https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Activation_Functions

Activation Function: Sigmoid

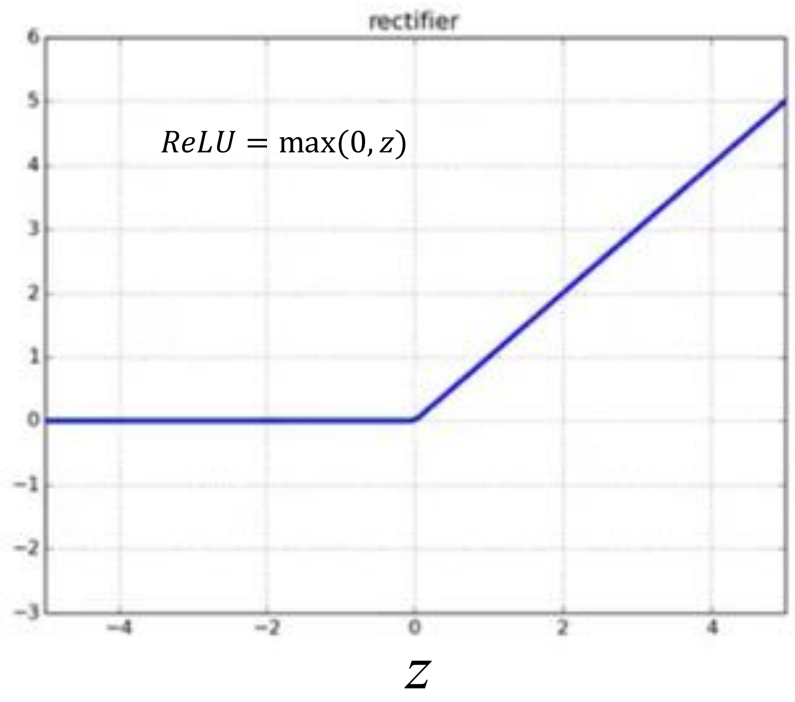


- The sigmoid function returns 0.5 when the input is 0.
- It returns a value close to 1 if the input is a large positive number.
- In the case of negative input, the sigmoid function outputs a value close to zero.

Activation Function: ReLU

https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

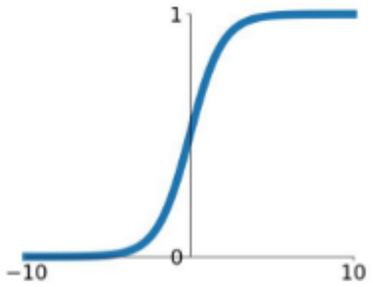
- The Rectified Linear Unit (ReLU) function computes the maximum of 0 and the input value z.
- ReLU is a piecewise linear function: if the input is positive, it outputs the input value; otherwise, it outputs zero.
- ReLU has emerged as the preferred activation function in many neural network architectures due to its ease of training and superior performance.



Activation Functions

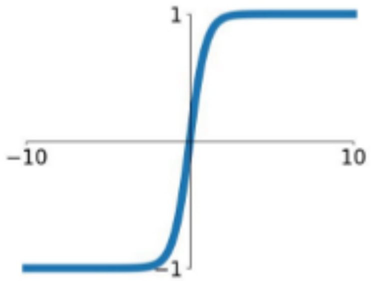
Sigmoid

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



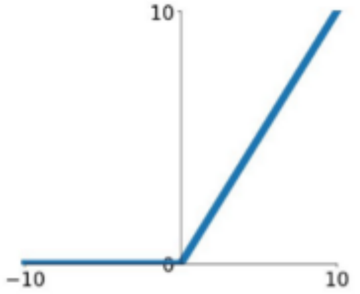
tanh

$\tanh(x)$



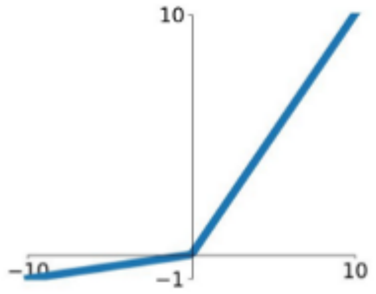
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

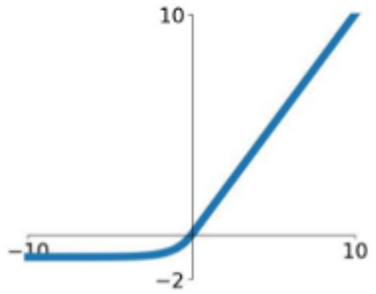


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Theory of ANN

- An artificial neural network is a supervised learning algorithm which means that we provide it the input data containing the independent variables and the output data that contains the dependent variable.
- For instance, in our example our independent variables are **X** = [x_1, x_2, x_3]
- The dependent variable is Y.
- Thus **X** -> Y, where **X** is a vector of data or features, and Y is a label or target variable
- In the beginning, the ANN makes some random predictions, these predictions are compared with the correct output and the error (the difference between the predicted values and the actual values) is calculated.

Theory of ANN

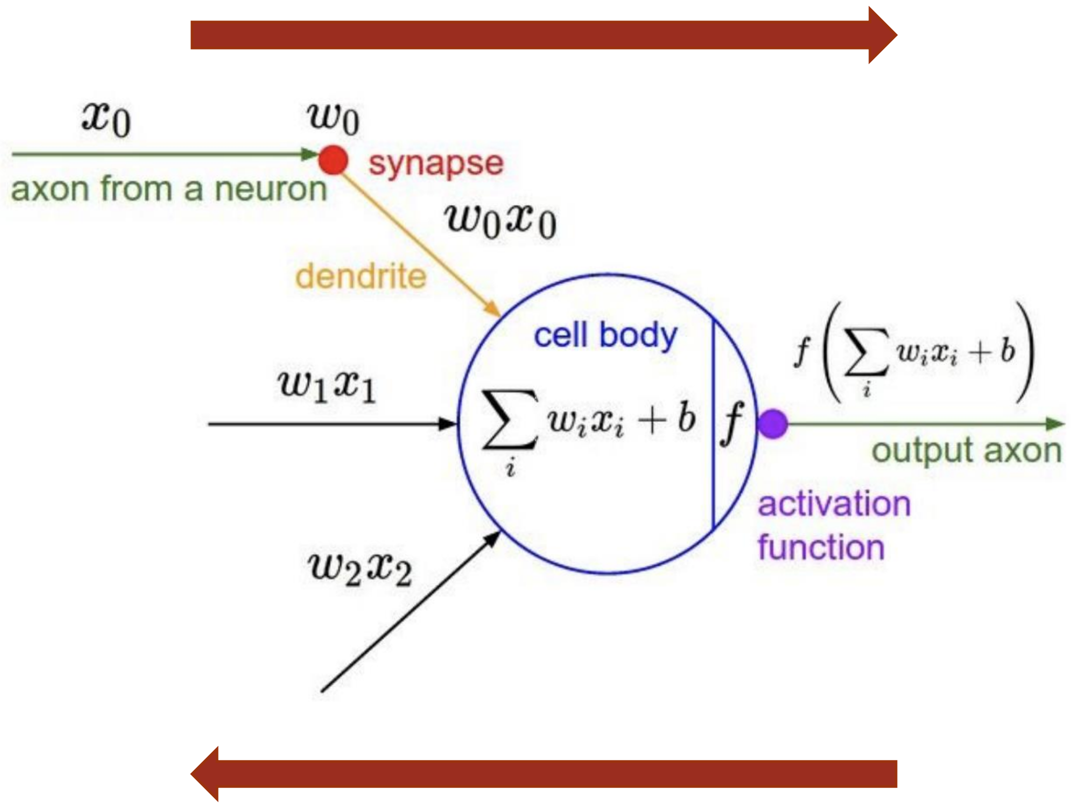
Simple Activation Function:

$$\Theta\left(\sum_{i=1}^{m}[w_i x_i] - b\right)$$

Unit Step Function

- The function that finds the difference between the actual value and the propagated values is called the cost function.
- The cost here refers to the error.
- Our objective is to minimize the error or cost function.
- Training a neural network basically refers to minimizing the cost function.
- We will see how we can perform this task.
- A neural network executes in two phases: Feed Forward phase and Back Propagation phase.

Feed Forward Direction

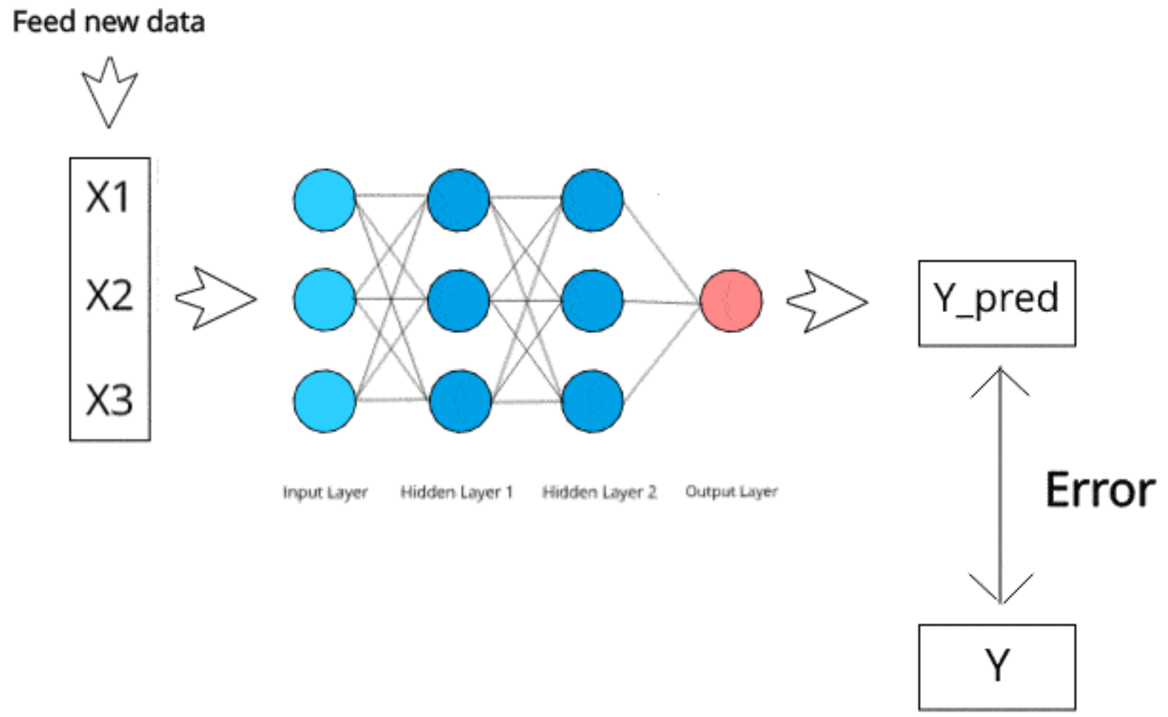


Back Propagation Direction

Backpropagation to Adjust Weights

There are basically 2 ways to adjust weights:

- Brute-force method
- Gradient descent



Training the ANN

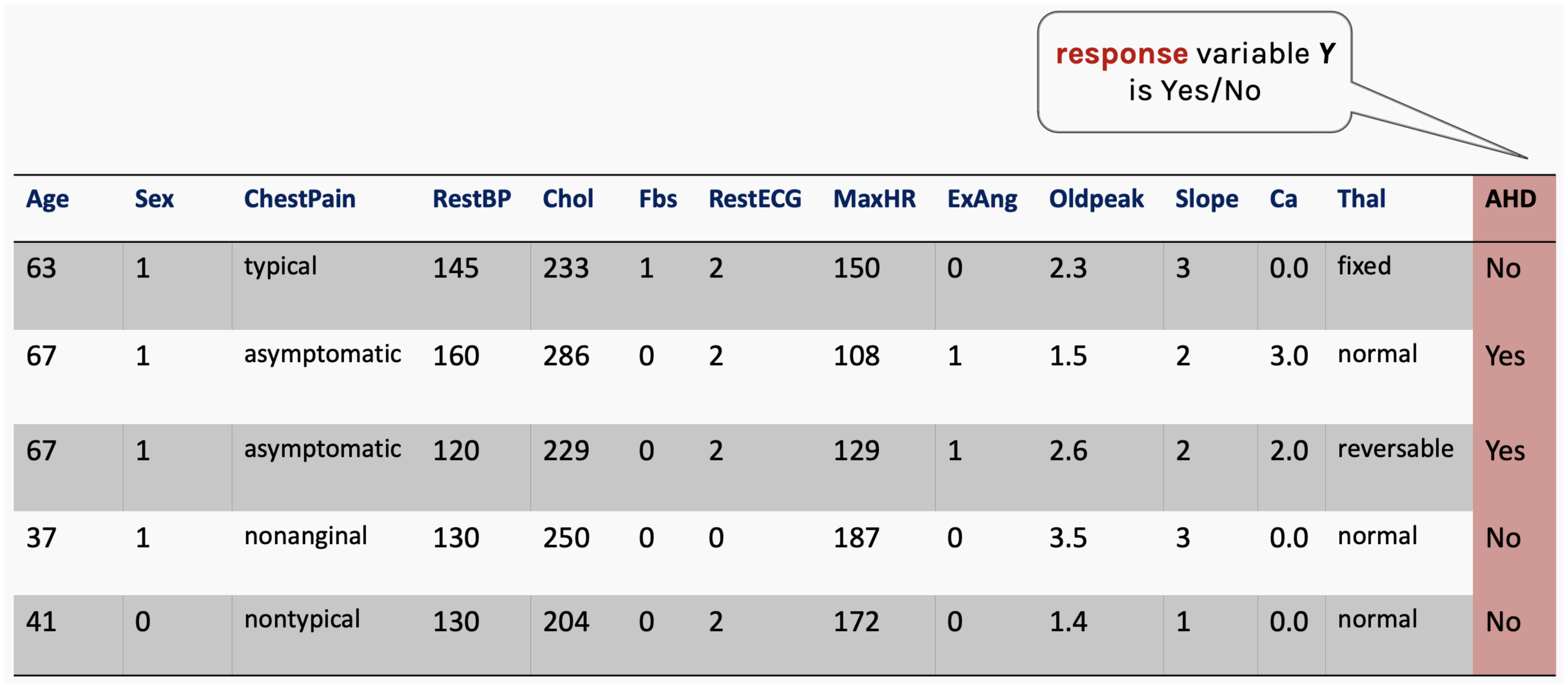
Basic Methods include 1) Gradient Descent and 2) Back Propagation

- To reiterate: Neural networks learn (or are trained) by processing examples, each of which contains a known "input" and "result," forming probability-weighted associations between the two, which are stored within the data structure of the net itself.
- The training of a neural network from a given example is usually conducted by determining the difference between the processed output of the network (often a prediction) and a target output.
- This difference is the error.
- The network then adjusts its weighted associations according to a learning rule using this error value.
- Successive adjustments will cause the neural network to produce output which is increasingly similar to the target output.
- After a sufficient number of these adjustments the training can be terminated based upon certain criteria.
- As you recall, this is known as supervised learning.

Classification

- Let us consider a binary classification problem.
- The goal is to attempt to classify each observation into a category (such as a class or cluster) defined by ***Y***, based on a set of feature variables ***X***.
- Let's say that we would like to predict whether a patient has heart disease based on features about the patient.
- The response variable here is categorical (binary), i.e., there are finite outcomes since there are only two categories (yes/no).

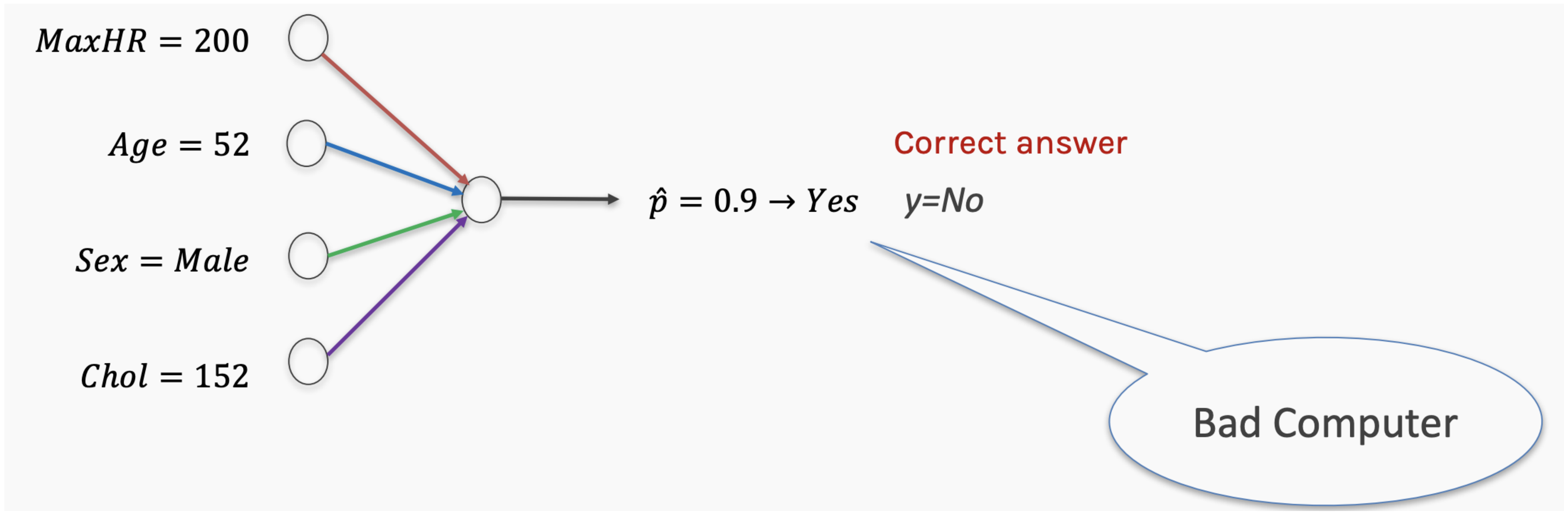
Example of heart disease: A categorical problem



response variable ***Y*** is Yes/No

Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

The Idea Behind Training an ANN for this problem

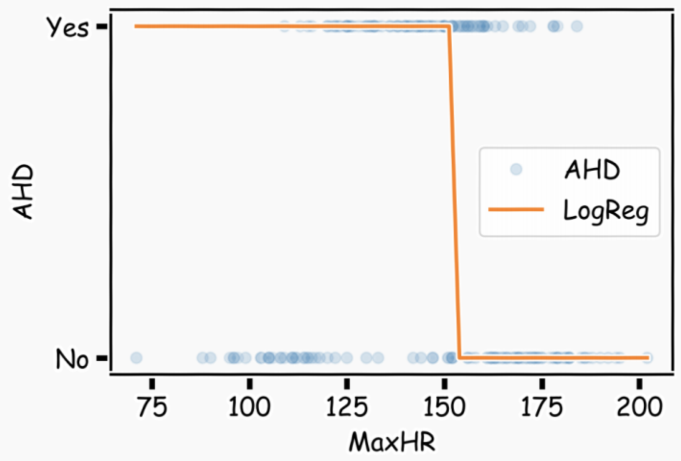


- The first thing we do is set randomly selected weights.
- Most likely it will perform horribly — in our heart data, the model will give us the wrong answer.
- We then ‘train’ the network by essentially punishing it for performing poorly.

Neural Networks are Often Trained by Gradient Descent

Likelihood Function:

$$\mathcal{L}(\beta_0, \beta_1) = -\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$



A likelihood function is combination of probabilities:

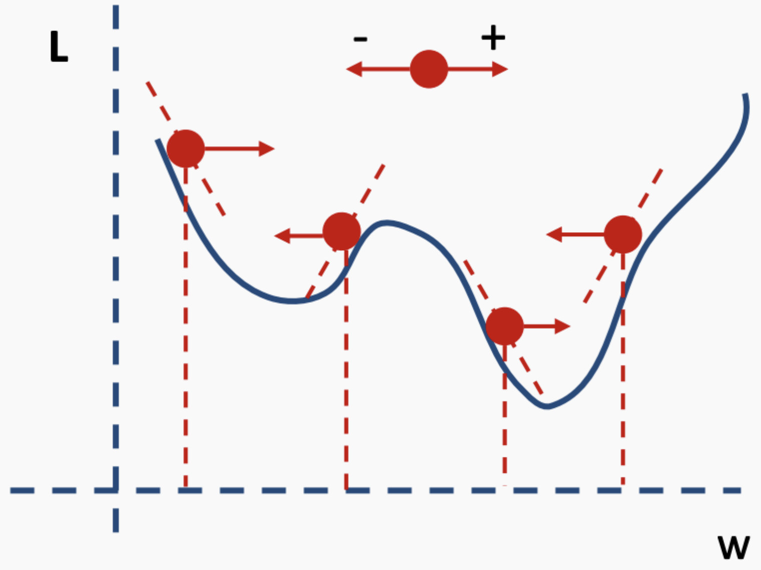
Label y_i = 1 (Yes) is produced with probability p_i

Label y_i = 0 (No) is produced with probability 1 - p_i

Gradient descent is basically Newtons' method

$$w^{(i+1)} = w^{(i)} - \underset{\uparrow}{\lambda} \frac{d\mathcal{L}}{dw}$$

Learning rate



Here we assume only 1 parameter is important

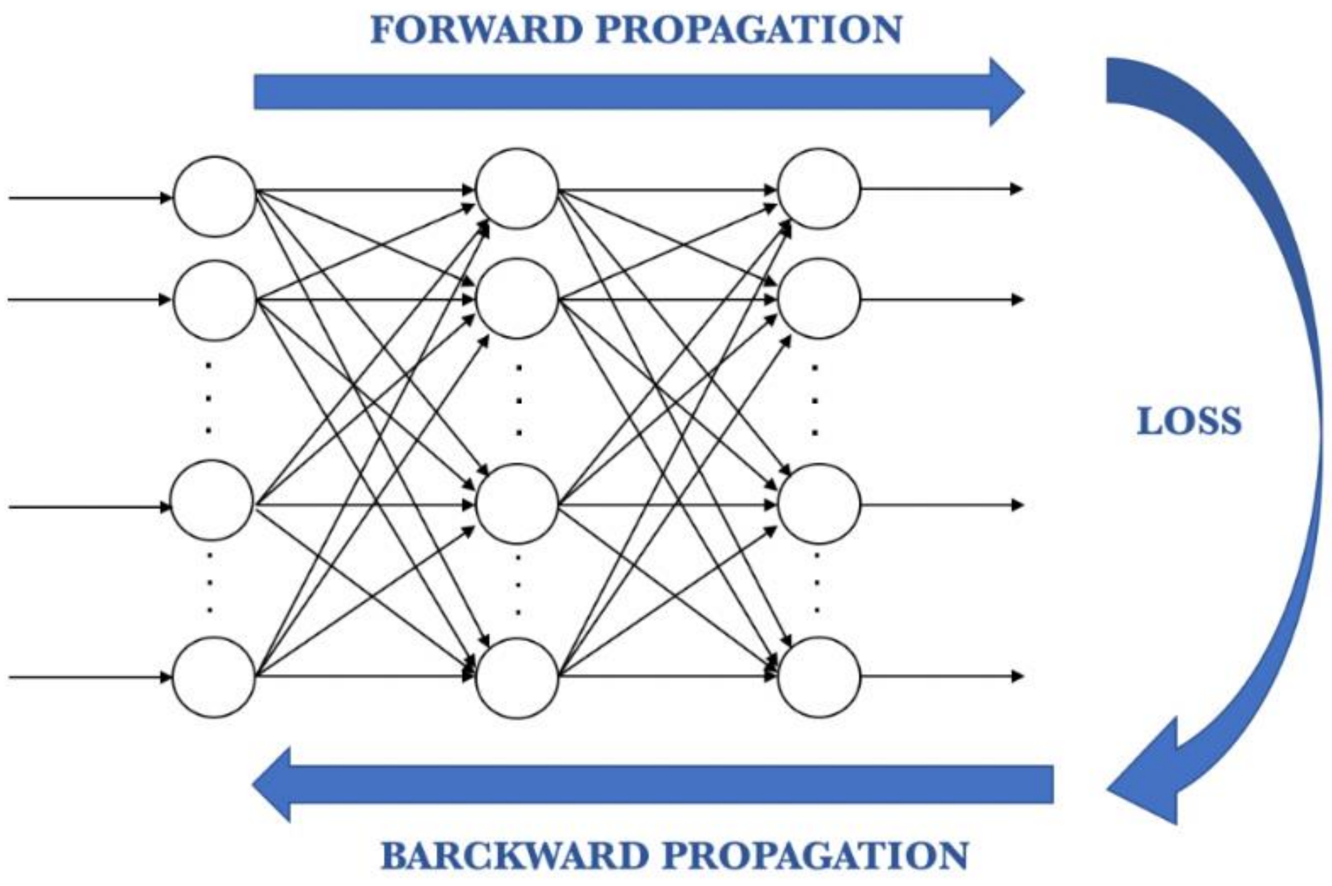
What about Multiparameter ANNs?

- What we just did mirrors the process used by neural network algorithms.
- Our example focused on using one feature for our model, but neural networks can handle multiple features.
- In a neural network, each feature has associated weights and there's also a bias term, collectively forming the regression parameters.
- The formulation varies slightly depending on whether the problem is a classification or regression problem.

Back Propagation is the General Training Method

- In the beginning, before you do any training, the neural network makes random predictions which are of course incorrect.
- We start by letting the network make random output predictions.
- We then compare the predicted output of the neural network with the actual output.
- We then calculate the error or loss function
- Next, we update the weights and the bias in such a manner that our predicted output comes closer to the actual output.
- In this phase, we can train our algorithm via *backpropgation*.
- Let’s take a look at the steps involved in the backpropagation phase.

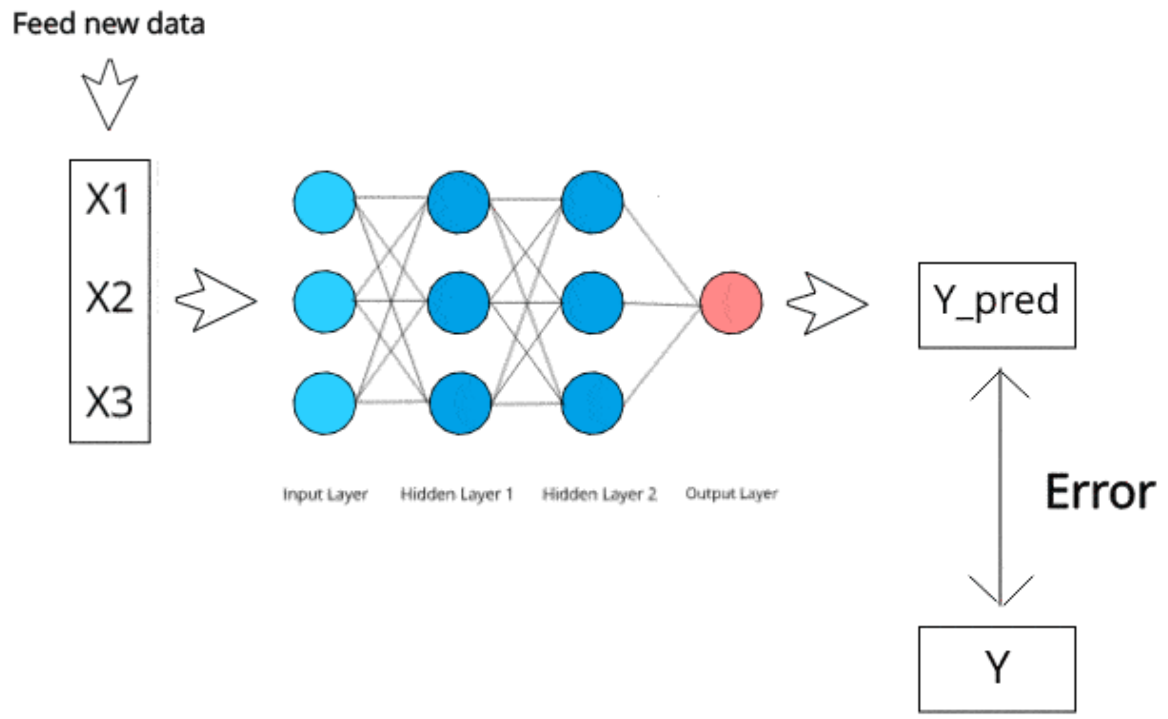
Training an ANN by Backpropagation



Backpropagation to Adjust Weights

There are basically 2 ways to adjust weights:

1. Brute-force method
2. Batch-Gradient descent

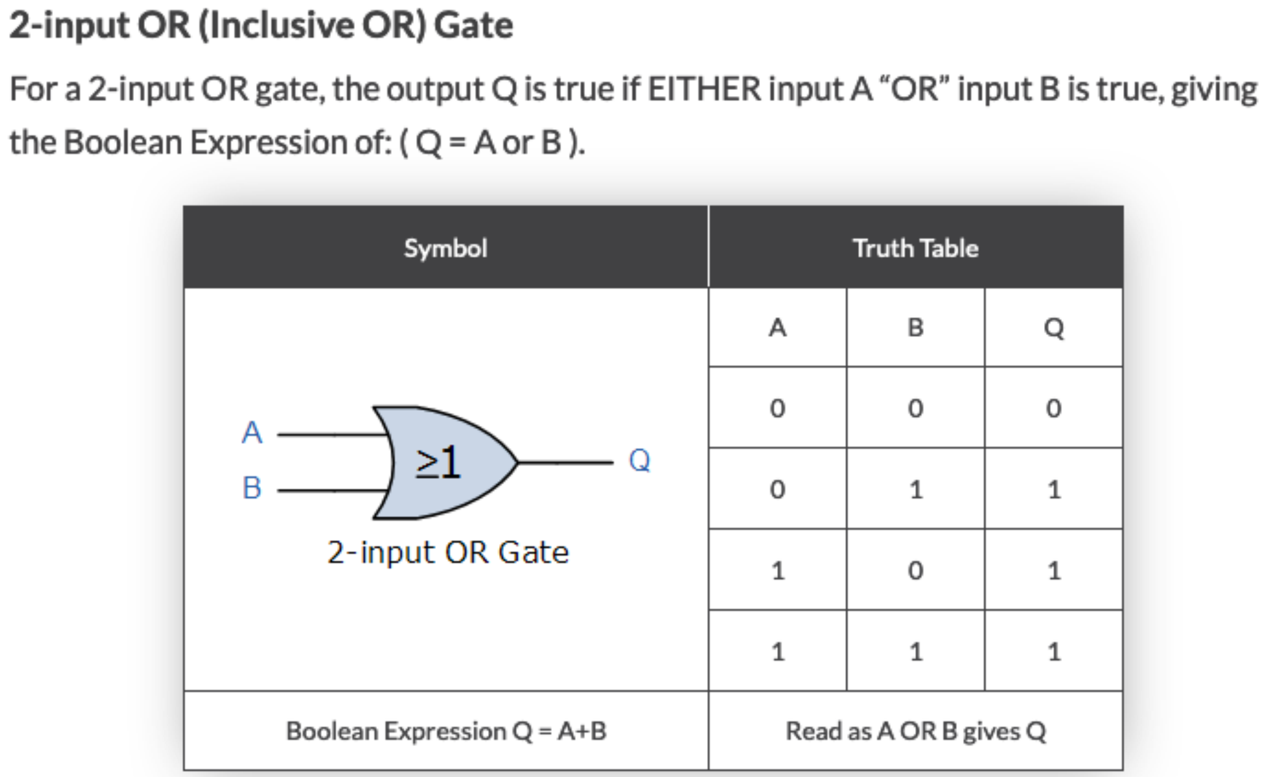


Simple Example of Back Propagation (Truth Table)

https://www.electronics-tutorials.ws/boolean/bool_7.html

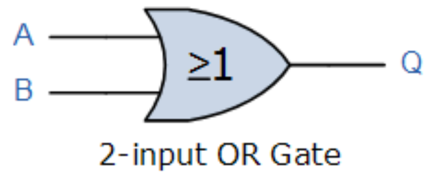
We will use as our activation function the threshold function $\Theta(x)$:

If x < 0, $\Theta(x) = 0.$ If x ≥ 0, $\Theta(x) = 1.$



2-input OR (Inclusive OR) Gate

For a 2-input OR gate, the output Q is true if EITHER input A "OR" input B is true, giving the Boolean Expression of: (Q = A or B).

Symbol	Truth Table		
	A	B	Q
 2-input OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression Q = A+B	Read as A OR B gives Q		

Truth tables are encoded in computers as logic gates

Truth Table Example of Backpropagation

https://users.math.yale.edu/public_html/People/frame/Fractals/

- Consider an OR statement (OR gate)

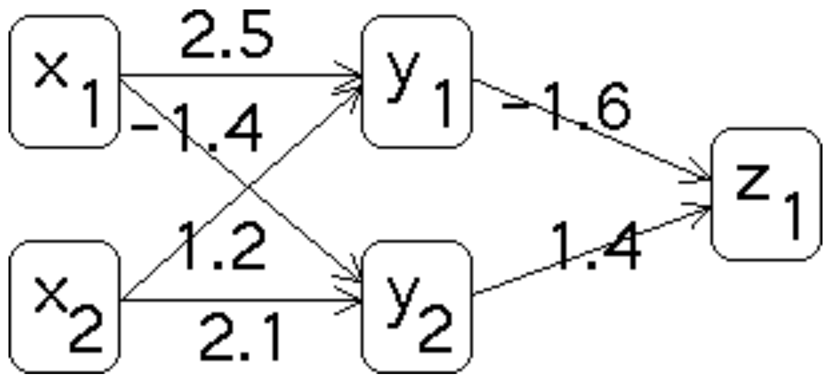
If x_1 OR x_2 = 1, then z_1 = 1

- We can set this problem up as a neural network.
- Initial (random) weights are as shown in the figure at right

For y_1 the input is: $2.5*X_1 + 1.2*X_2 =$
$2.5*0 + 1.2*1 = 1.2$

For y_2 the input is: $-1.4*X_1 + 2.1*X_2 =$
$-1.4*0 + 2.1*1 = 2.1$

We have assumed $X_1 = 0$ and $X_2 = 1$



- Using a threshold function with a threshold value of 0, we see outputs $y_1 = 1$ and $y_2 = 1$.
- Now the weighted input for the output neuron is:

$-1.6*Y_1 + 1.4*Y_2 = -1.6*1 + 1.4*1 = -0.2$

Consequently, upon applying the threshold function, the output value is $z_1 = 0$ when it should be 1

Threshold function $\Theta(x)$: If $x < 0$, $\Theta(x) = 0.$ If $x \geq 0$, $\Theta(x) = 1.$

⬇ = Weights

Training the Neural Net for the OR Gate

- For example, again suppose the training set for the net pictured previously contains the input-output pair:

$x_1 = 0$

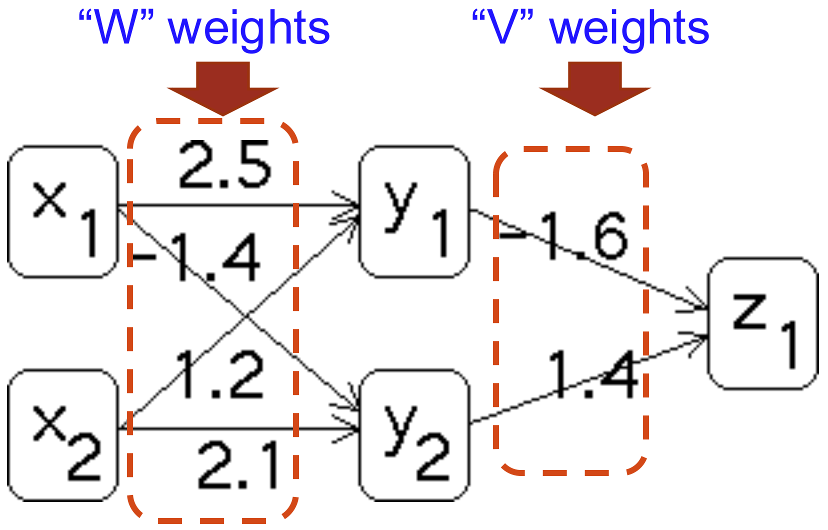
$x_2 = 1$

$\Rightarrow Z_1 = 1$

The net pictured at right is not trained for this input-output pair.

- As we showed in the previous slide, the output error is:

$e_{z1} = 1 - 0 = 1$



Nomenclature:

- The 4 weights connecting an x_i node and the y_j node are designated by "w".
- The 2 weights connecting a y_j node with output node z is designated by "v"

Training by Backpropagation (continued)

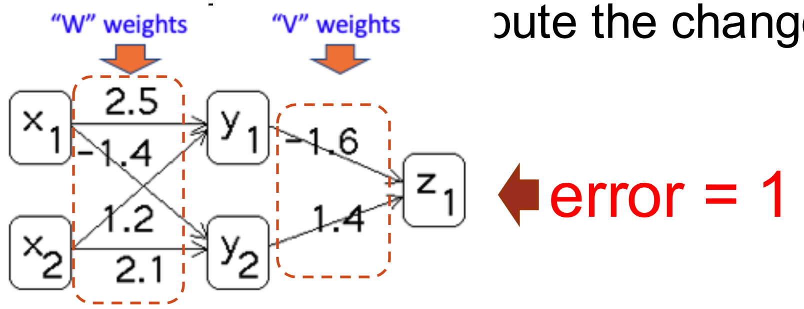
- We now "backpropagate" the errors, which are generally computed at a neuron as the "upstream" error times the existing weight
- The weight $v_{1,1}$ gives the error at y_1:

$$e_{y1} = e_{z1} * v_{1,1} = 1*(-1.6) = -1.6$$

- Similarly, the weight $v_{2,1}$ gives the error at y_2:

$$e_{y2} = e_{z1} * v_{2,1} = 1*1.4 = 1.4$$

With the pute the changes in the we



$$\Delta v_{1,1} = e_{z1} * y_1 = 1*1 = 1$$

$$\Delta v_{2,1} = e_{z1} * y_2 = 1*1 = 1$$

$$\Delta w_{1,1} = e_{y1} * x_1 = -1.6*0 = 0$$

$$\Delta w_{2,1} = e_{y1} * x_2 = -1.6*1 = -1.6$$

$$\Delta w_{1,2} = e_{y2} * x_1 = 1.4*0 = 0$$

$$\Delta w_{2,2} = e_{y2} * x_2 = 1.4*1 = 1.4$$

The corrections to the weights are computed as the error e times the existing output at the "downstream" neuron, i.e.: e times (1 or 0)

Now new weights are computed.

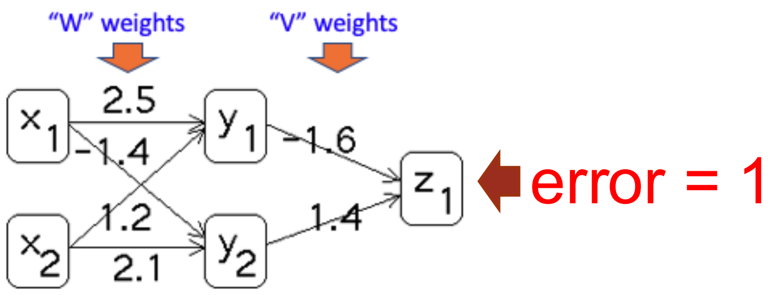
$$v_{1,1} \to v_{1,1} + \Delta v_{1,1} = -1.6 + 1 = -0.6$$
$$v_{2,1} \to v_{2,1} + \Delta v_{2,1} = 1.4 + 1 = 2.4$$
$$w_{1,1} \to w_{1,1} + \Delta w_{1,1} = 2.5 + 0 = 2.5$$
$$w_{2,1} \to w_{2,1} + \Delta w_{2,1} = 1.2 + (-1.6) = -0.4$$
$$w_{1,2} \to w_{1,2} + \Delta w_{1,2} = -1.4 + 0 = -1.4$$
$$w_{2,2} \to w_{2,2} + \Delta w_{2,2} = 2.1 + 1.4 = 3.5$$

Note: Error $e_{z1} = 1$; $y_1 = y_2 = 1$; $x_1 = 0$; $x_2 = 1$

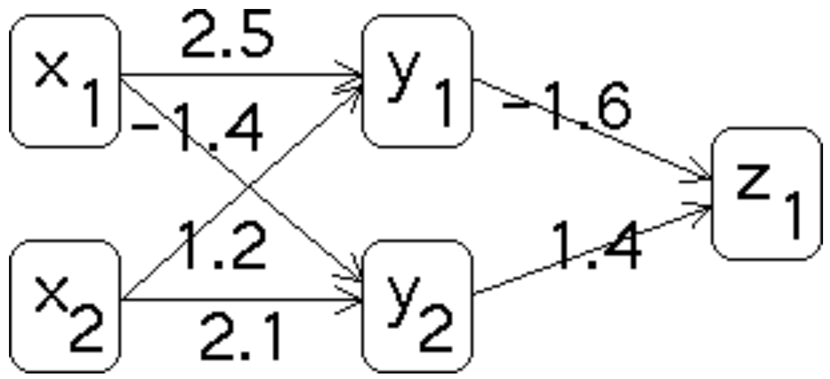
Training (continued)

- Here is the new neural net.
- It is easy to verify that the output is equal to $z_1 = 1$ as needed, given inputs of:
  - $x_1 = 0$
  - $x_2 = 1$

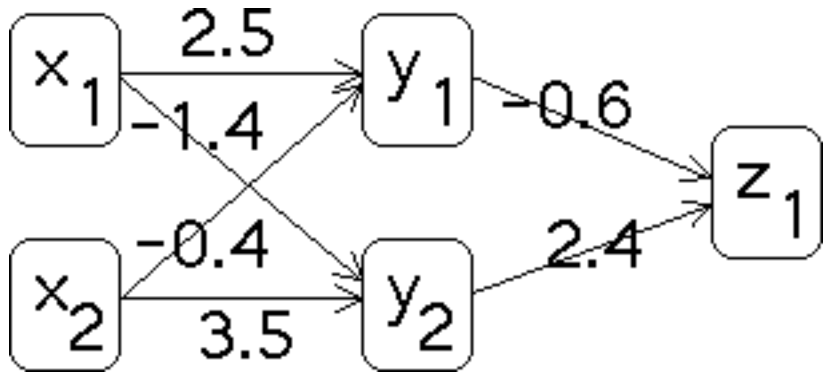
Note that in back propagation, the error is "propagated back" through the net, using the weights to compute errors at the hidden neurons, and ultimately to adjust the weights.



Original Net

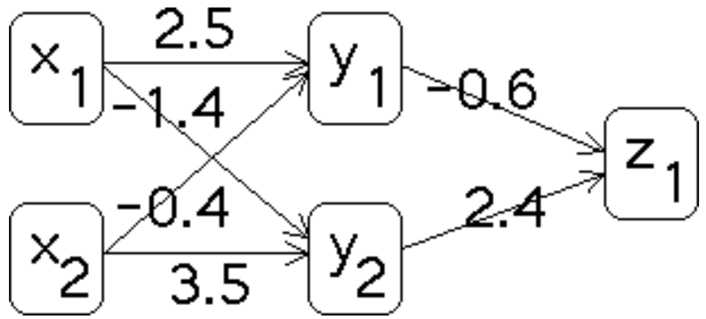


Trained Net



Truth Table for the Trained Net

https://users.math.yale.edu/public_html/People/frame/Fractals/



- Consider an OR statement (OR gate)

If x_1 OR x_2 = 1, then z_1 = 1

- We can set this problem up as a neural network.
- Initial (random) weights are as shown in the figure at right

For y_1 the input is: $2.5 \cdot X_1 - 0.4 \cdot X_2 =$
$2.5 \cdot 0 - 0.4 \cdot 1 = 2.1$

For y_2 the input is: $-1.4 \cdot X_1 + 3.5 \cdot X_2 =$
$-1.4 \cdot 0 + 3.5 \cdot 1 = 3.5$

We assume $X_1 = 0$ and $X_2 = 1$

- Using a threshold function with a threshold value of 0, we see outputs $y_1 = 1$ and $y_2 = 1$.
- Now the weighted input for the output neuron is:

$-0.6 \cdot Y_1 + 2.4 \cdot Y_2 = -0.6 \cdot 1 + 3.5 \cdot 1 = 2.9$

Consequently, upon applying the threshold function, the output value is $z_1 = 1$ in agreement with the correct value = 1

Threshold function $\Theta(x)$: If $x < 0$, $\Theta(x) = 0.$ If $x \geq 0$, $\Theta(x) = 1.$

= Weights

Uses of Artificial Neural Networks

- Deep Learning models can be used for a variety of complex tasks:
  - Artificial Neural Networks(ANN) for Regression and classification
  - Convolutional Neural Networks(CNN) for Computer Vision
  - Recurrent Neural Networks(RNN) for Time Series analysis
  - Self-organizing maps for Feature extraction
  - Deep Boltzmann machines for Recommendation systems
  - Auto Encoders for Recommendation systems
- A typical example is Google Translate
- Refer to the TensorFlow playground on the web that allows one to experiment with ANNs in an interactive setting

TensorFlow & useful t

To the tutorials

Keras is most suitable for:

- Rapid Prototyping
- Small Dataset
- Multiple back-end support

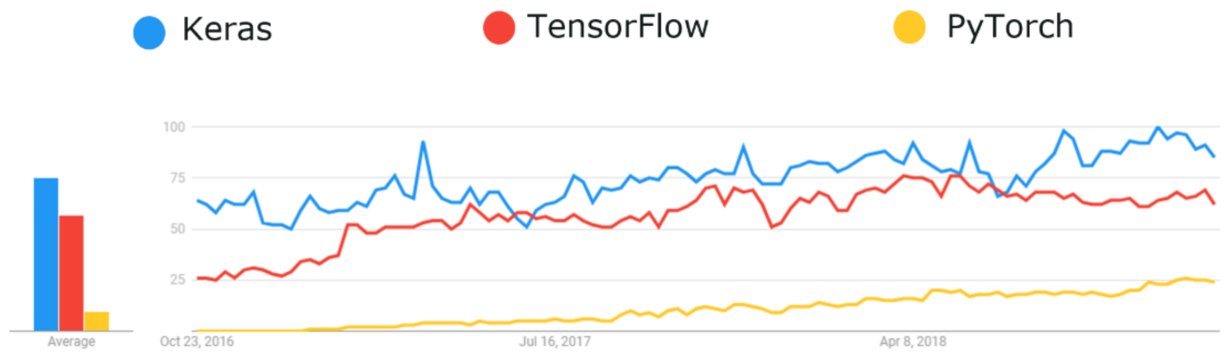
TensorFlow is most suitable for:

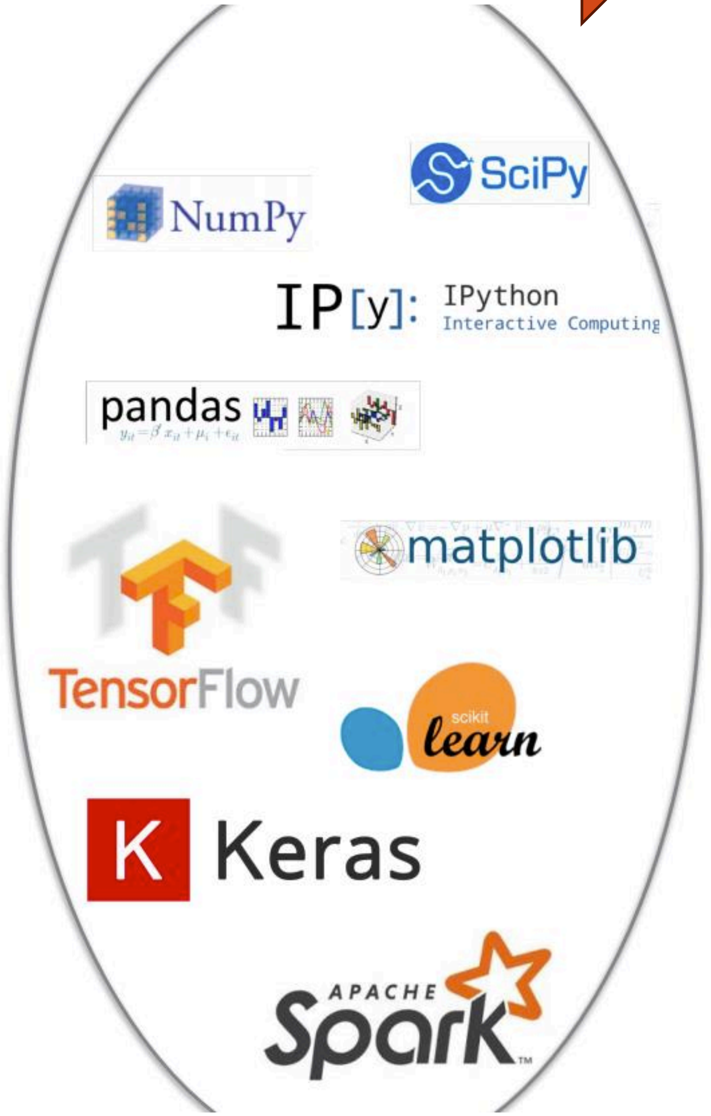
- Large Dataset
- High Performance
- Functionality
- Object Detection

PyTorch is most suitable for:

- Flexibility
- Short Training Duration
- Debugging capabilities

Popularity





ML in Physics

Condensed Matter Physics:

Phase Transition Detection: Machine learning models, such as neural networks, are used to identify phase transitions in materials by analyzing large datasets of configurations or states without explicit knowledge of the underlying physical model.

Particle Physics:

Event Classification in Collider Experiments: Machine learning is heavily used at CERN in experiments like the Large Hadron Collider (LHC) to classify events, distinguish signal from background, and optimize data analysis pipelines.

Astrophysics:

Exoplanet Detection: Machine learning algorithms are employed to analyze data from telescopes (like the Kepler mission) to detect exoplanets by identifying subtle patterns in light curves.

Quantum Physics:

Quantum State Tomography: Machine learning techniques are applied to reconstruct quantum states more efficiently and accurately than traditional methods, especially for systems with large Hilbert spaces.

Plasma Physics:

Fusion Reactor Control: In fusion research, machine learning is used for real-time control of plasma in tokamaks, predicting disruptions, and optimizing plasma confinement.

Biophysics:

Protein Folding Prediction: Deep learning models, such as AlphaFold, have made significant advances in predicting protein structures based on their amino acid sequences, addressing long-standing problems in biophysics.

High-Energy Physics:

Jet Tagging: Machine learning is used to classify and identify jets in high-energy particle collisions, helping to better understand the fundamental particles and their interactions.

Geophysics:

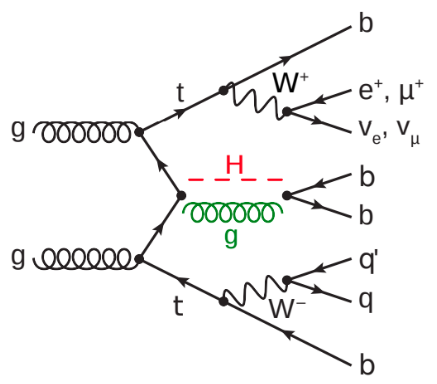
Earthquake Prediction: Machine learning is used to analyze seismic data for predicting the occurrence of earthquakes, identifying patterns in data that traditional methods might miss.

Classification

Higgs classification $t\bar{t}H(b\bar{b})$

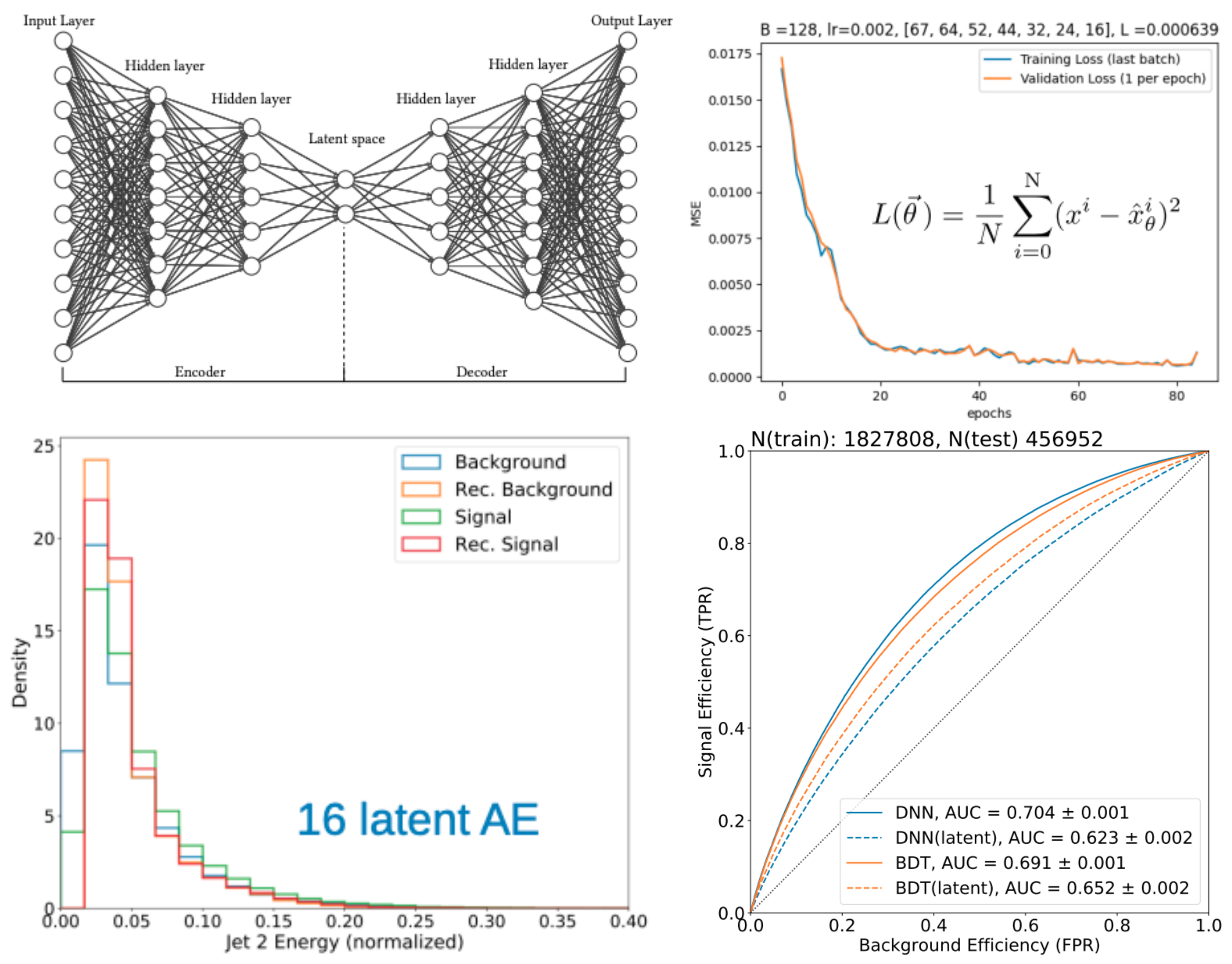
C. Reissel, ML4Jets 2021
V. Belis, arxiv: 2104.07692

- Ho to identify/sperate higgs boson from the irreducible background



- 67 observables that are used to discriminat signal from background (**chanllange!**)

Auto-Encoders (AE) for Feature reduction

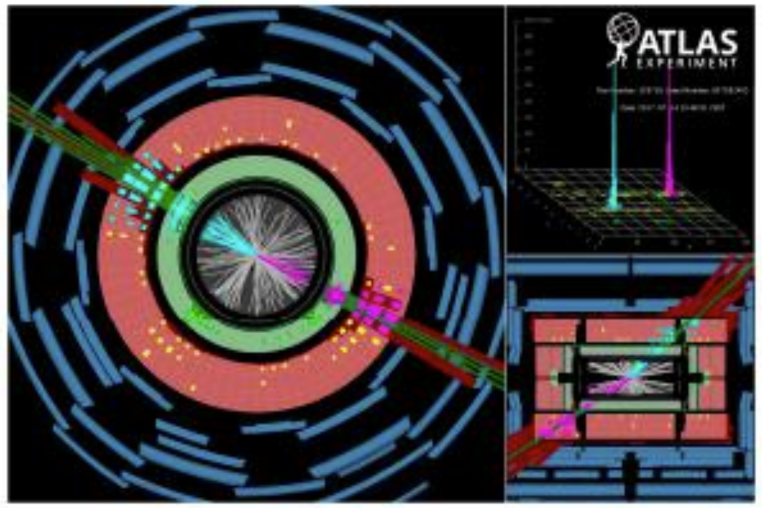


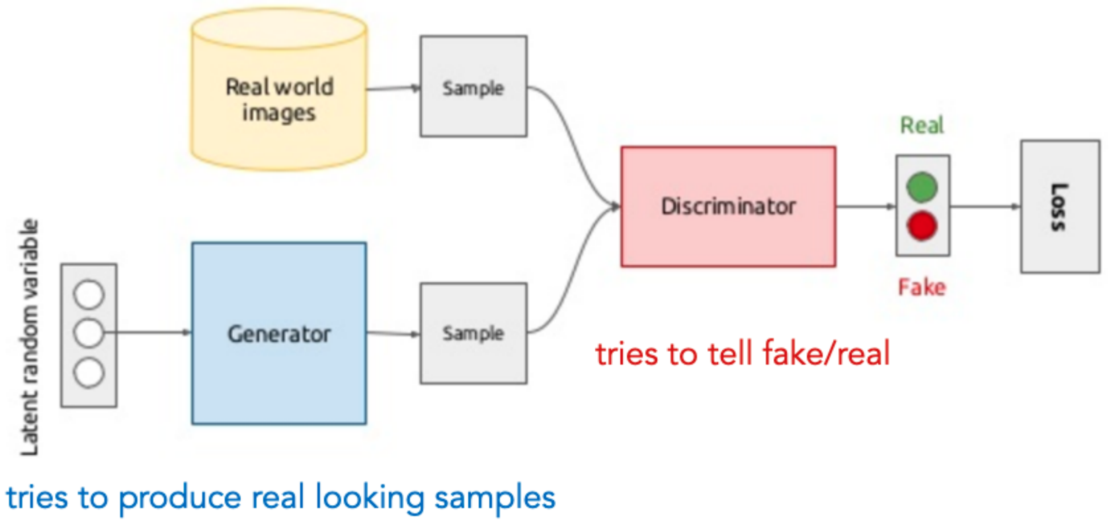
Event simulations

GANs

Michael Kagan SLAC

- GANs = Generative Models approximate and simulate the data generation process
- Scientific Simulators
  - Built from science knowledge
  - Relatively few parameters, often interpretable
  - High Fidelity, often computationally costly
- Machine Learning GANs
  - Fit to data, inductive bias in model design / optimization
  - Can have >10^6 parameters, often not interpretable
  - Often slow to train, fast to evaluate

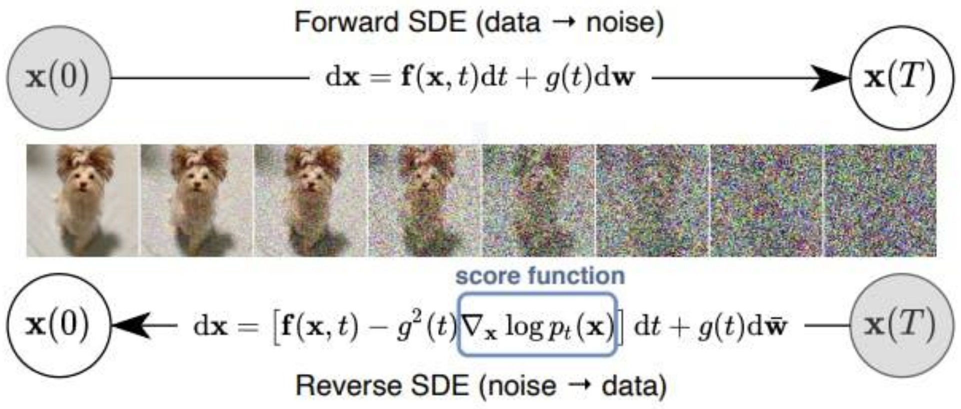




Diffusion models

Generative model

- Adding noise to the training data and then learning to remove the noise to recover the original data.
- A diffusion model simulates the natural process of diffusion, where particles or substances spread out from areas of high concentration to areas of low concentration over time
- In machine learning, this process is applied to data, such as images or text, to gradually transform them into pure noise.
- Two phases:
  - The forward diffusion process adds noise to the data in a series of steps, each step increasing the noise level.
  - The reverse diffusion process removes noise from the data in a series of steps, each step decreasing the noise level.



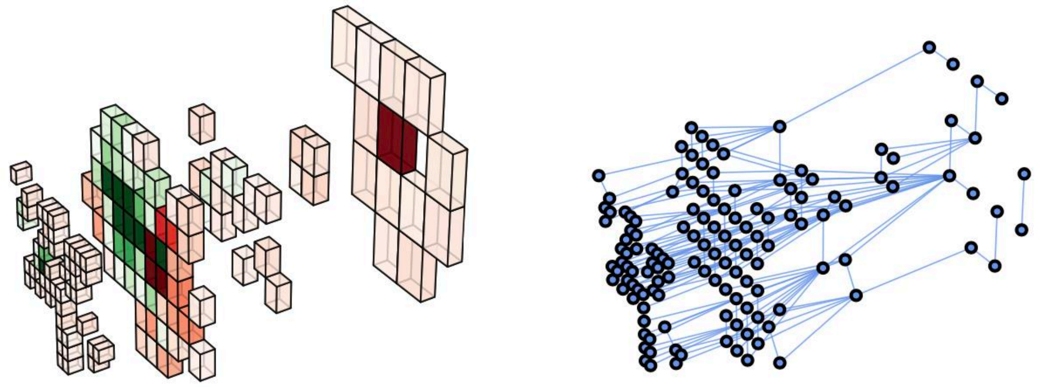
Forward SDE (data → noise)
$\mathbf{x}(0)$
$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}]dt + g(t)d\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

- Learn by the feedback it receives during the reverse diffusion process.
  - The feedback is usually the difference between the noisy data and the original data at each step.
  - Can generate new data samples by passing randomly sampled noise through the reverse diffusion process.
  - Benefit from physical insights, principles, and models, such as non-equilibrium thermodynamics, Markov chains, and Langevin dynamics.

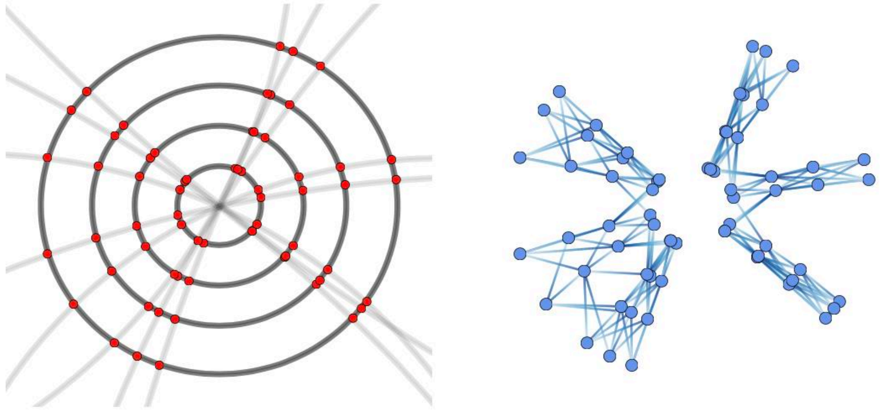
Graph NNs for particle tracking

- Graphs are the most natural and powerful way of representing unstructured such as this of particle tracking
- GNN’s is a geometric deep learning architecture that implements learning functions to operate on graphs with relational inductive bias
- A message-passing interface is performed whereby information is propagated across various nodes and edges in the graph allowing essential connections to originate and edge-, node-, and graph-level outputs to be computed

neighbouring energy deposits in calorimeter cells



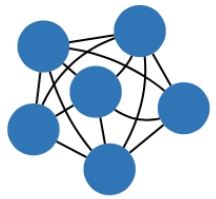
segments of hits in a tracking detector hits

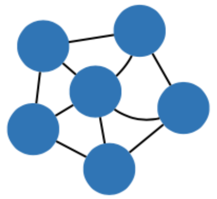


Graph NNs for particle tracking

Methods to construct graphs

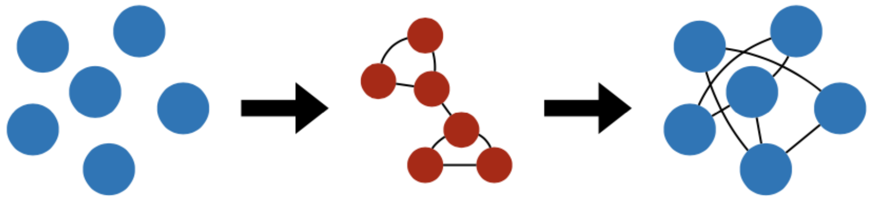
Connecting all pairs of nodes





Connecting neighbouring nodes

connecting neighbouring nodes in a latent feature space

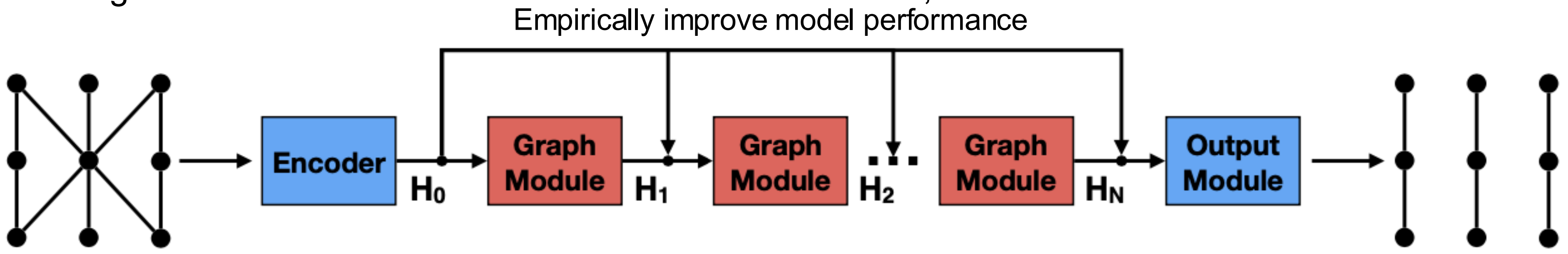


- It may be necessary to restrict the considered edges
- Edges can be formed based on the input features (e.g., the $\Delta R = \sqrt{\Delta\varphi^2 + \Delta\eta^2}$ between particles)
- Given a distance metric between nodes and a criterion for connecting them, such as k-nearest neighbours or a fixed maximum distance, the edges can be created

Graph NNs for particle tracking

Example arxiv: 2003.11603

- In the input graphs:
  - Node features are the three cylindrical coordinates (r, φ, z)
  - Edge features are the difference of the coordinates (Δη, Δφ)
  - The edge labels are 1 if two hits come from the same track, and 0 otherwise



- Encoder which transforms input node and edge features into their latent representations
- Graph module which performs message passing to update latent features
- Output module which computes edge classification scores
- With small modification in the same network can be used for energy clustering

Thank you

Contact

Ashraf Mohamed
University of Texas at Austin
E-mail: ashraf.mohamed@austin.utexas.edu