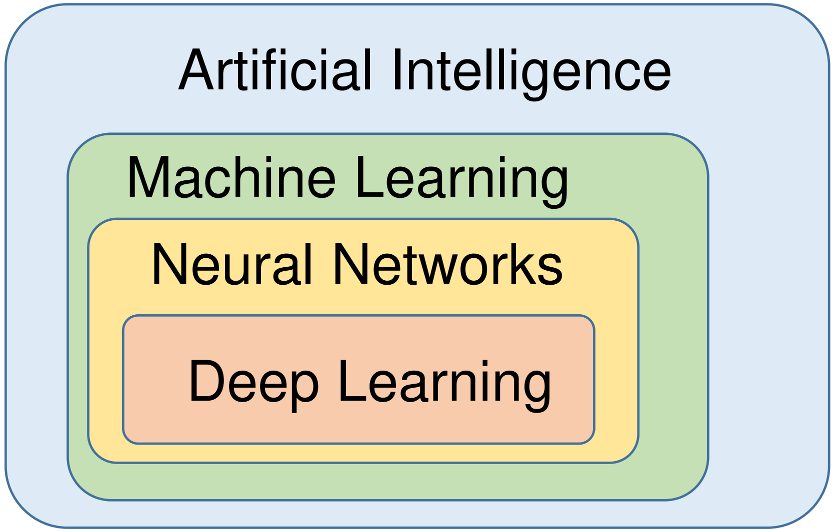
Artificial Intelligence (AI) and Machine Learning (ML)

AI: Computer programs that mimic human intelligence to perform complex tasks

ML: adaptable algorithms that trains their parameters from data

DL (Deep Learning): ANNs formed by several layers of artificial neurons

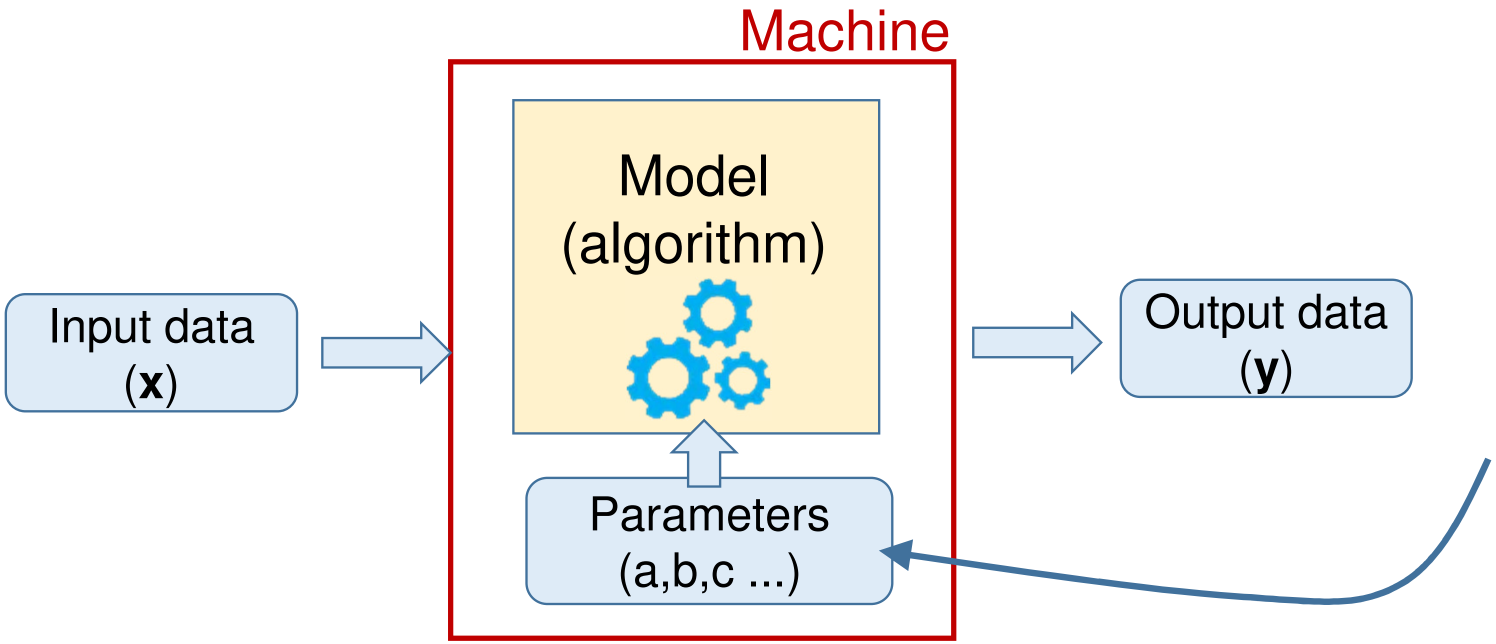


ANN (Artificial Neural Networks): ML algorithm based on a structure that resemble the neuron connections in the brain

Very short introduction to Machine Learning

Definition

- Machine Learning (ML) indicates all kinds of strategies and methods that allow tuning an artificial system ("the machine") to make predictions about <u>future data</u> the basis of <u>past data</u>.
- ML is all about "self-learning" algorithms.
- ML is subfield of Artificial Intelligence (AI).



In a <u>ML system</u>, the correct values of the parameters to obtain the desired response is autonomously determined by the machine after a <u>"training" phase</u>

Example of Applications of Machine Learning

- Every kind of feature recognition in images (faces, objects, hazards, etc.)
- Voice recognition (form sounds to words)
- Text interpretation (spam filters, web-search tasks, document selection ...)
- Pattern recognition from homogeneous or heterogeneous sensor data (chemical component recognition, medical diagnosis, food quality ...)

Advanced application in sensor systems

The three different cases of Machine Learning

1) Supervised learning ⇨ *Frequently used in sensor applications*

2) Unsupervised learning

3) Reinforcement learning

Supervised Learning

Single input data "<u>example</u>"

<u>Label</u> **y**

$$\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_k \end{pmatrix} \qquad \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_h \end{pmatrix}$$

Set of "<u>features</u>", or "predictors" that can be arranged into a row or column vector **x**

Data in this format are called "<u>labeled data</u>"

Training:

A large number of examples (labelled data) is proposed to the machine that learns to associate different labels (**y**) to different properties of the feature vector (**x**).

Goal:

Tune the algorithm (i.e. its parameters) is such a way that it gets able to <u>**predict**</u> the <u>label</u> of all new x feature vector that are proposed to it after training.

Supervised Learning

There are three main cases of supervised learning, depending on the type of label

Label

- Binary (only two values, e.g. "yes / not") ⇒ Binary classification
- Discrete (only a set of *n* possible values) ⇒ Multi-class classification
- Continuous (e.g. a real number) ⇒ Regression

These categories are independent of the type of input data (i.e. type of features), which can be continuous even in the case of binary or discrete label

More on classification

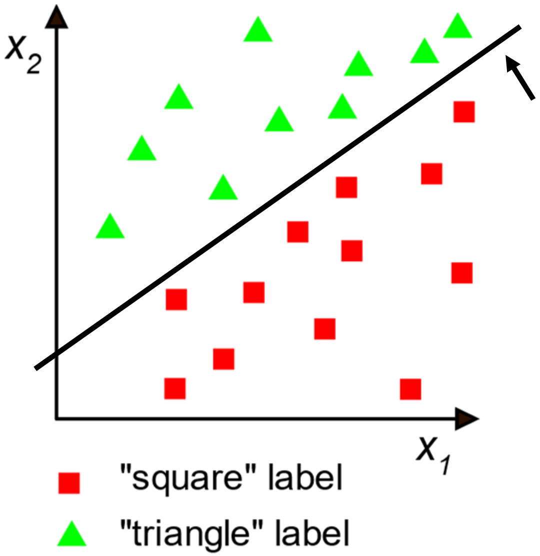
Non-probabilistic classification: the membership to a given class is predicted
Probabilistic classification: the probability of belonging to a class is predicted

Linear and non-linear classification

Linear

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Feature vector

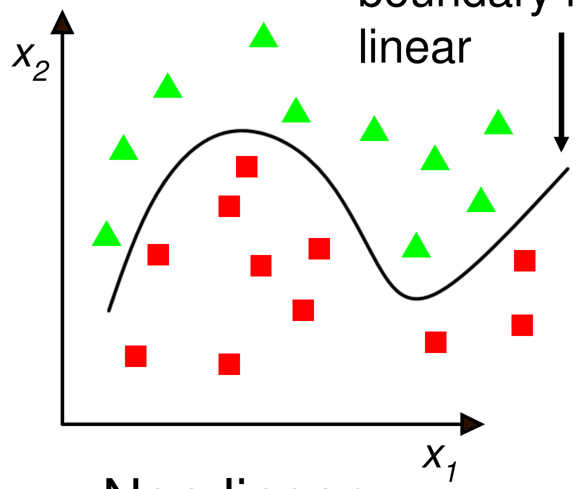


linear "decision boundary"

In more than 2 dimensions the linear boundary is a hyperplane

Graphical representation of the <u>training set</u>

The decision boundary is non-linear



Non-linear

Unsupervised learning

$$\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_k \end{pmatrix}$$

Single input data "example"

Data are "unlabeled"

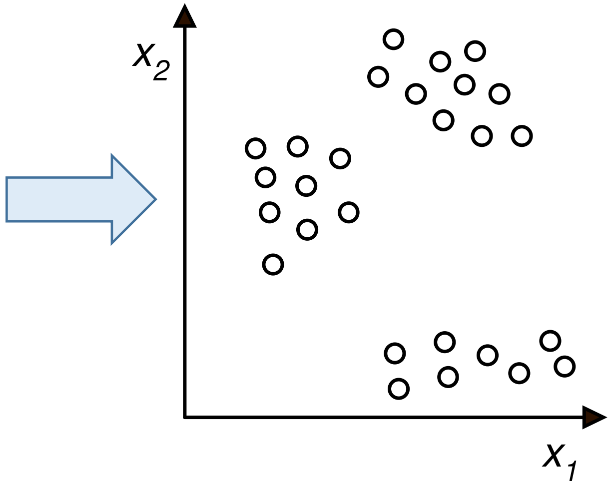
The goal of the training is finding hidden characteristics that can be used to find a criterion to classify the data (data clustering) or to eliminate useless information (dimensionality reduction)

To illustrate the typical goals of unsupervised learning it is better to show a few examples.

Example of Data Clustering

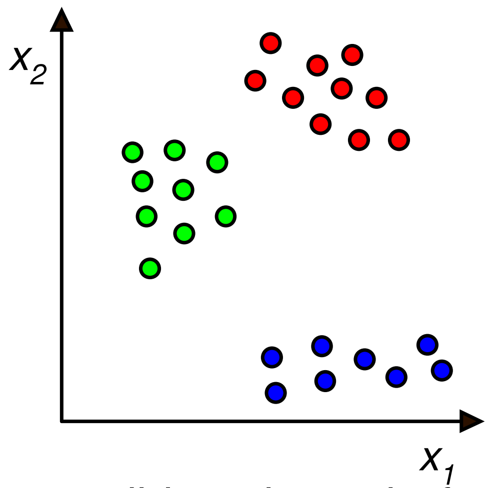
$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Feature vector



Graphical representation of the training set (27 examples)

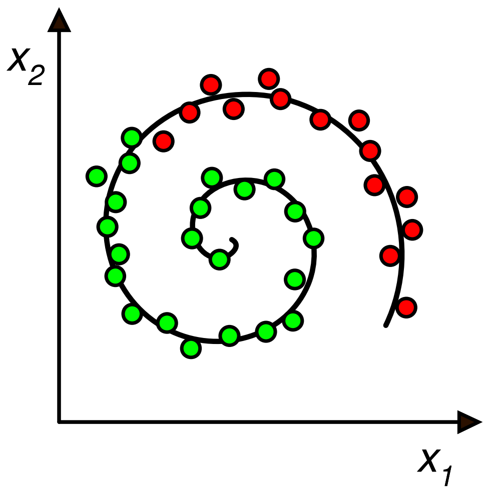
The clustering algorithm understands that data can be grouped into three "clusters". After clustering, a label can be created to signify belonging to a given cluster.



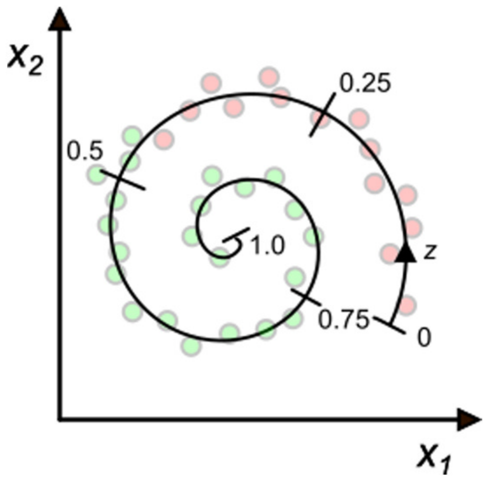
In this example the label can assume three values (red, green, blue)

Once we have learnt this property of the data (that we did not know before) we still have to give an interpretation to it and if it is useful, use this information to simplify classification algorithms.

Example of dimensionality reduction



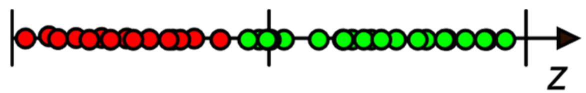
We have a set of labelled data (green / red) with a 2-D vector. Depending on the algorithm, classification can be difficult with data arranged in this way.



It is clearly possible to arrange data according their position in the spiral (coordinate z).
<u>For this operation, the algorithm do not use the label</u>

The dimensionality reduction task consists in finding coordinate transformation that allow reducing the number of significant features

With this transformation, classification is much easier

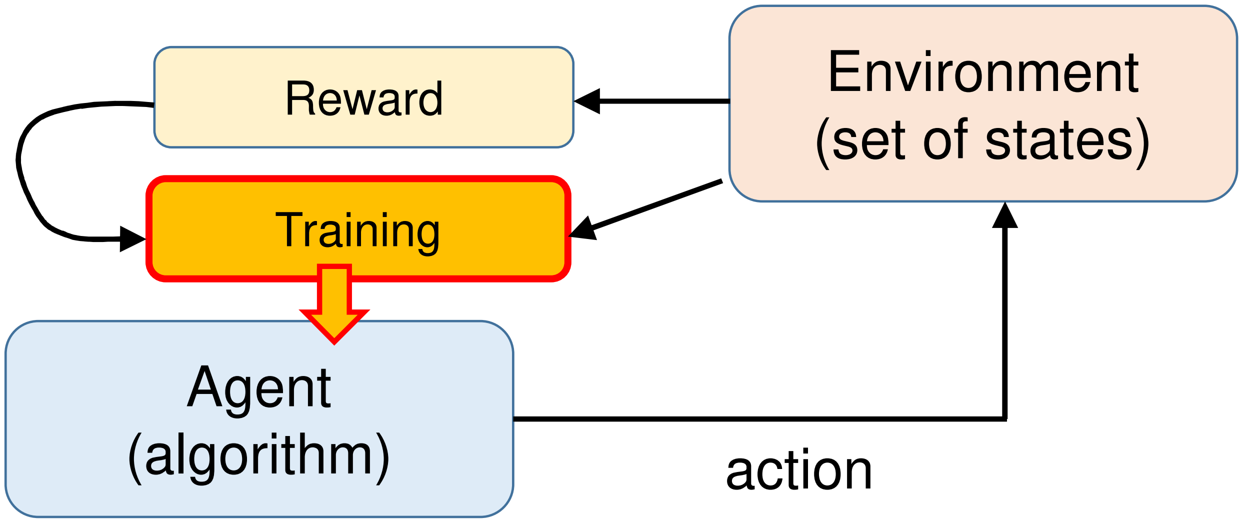


Reinforcement learning

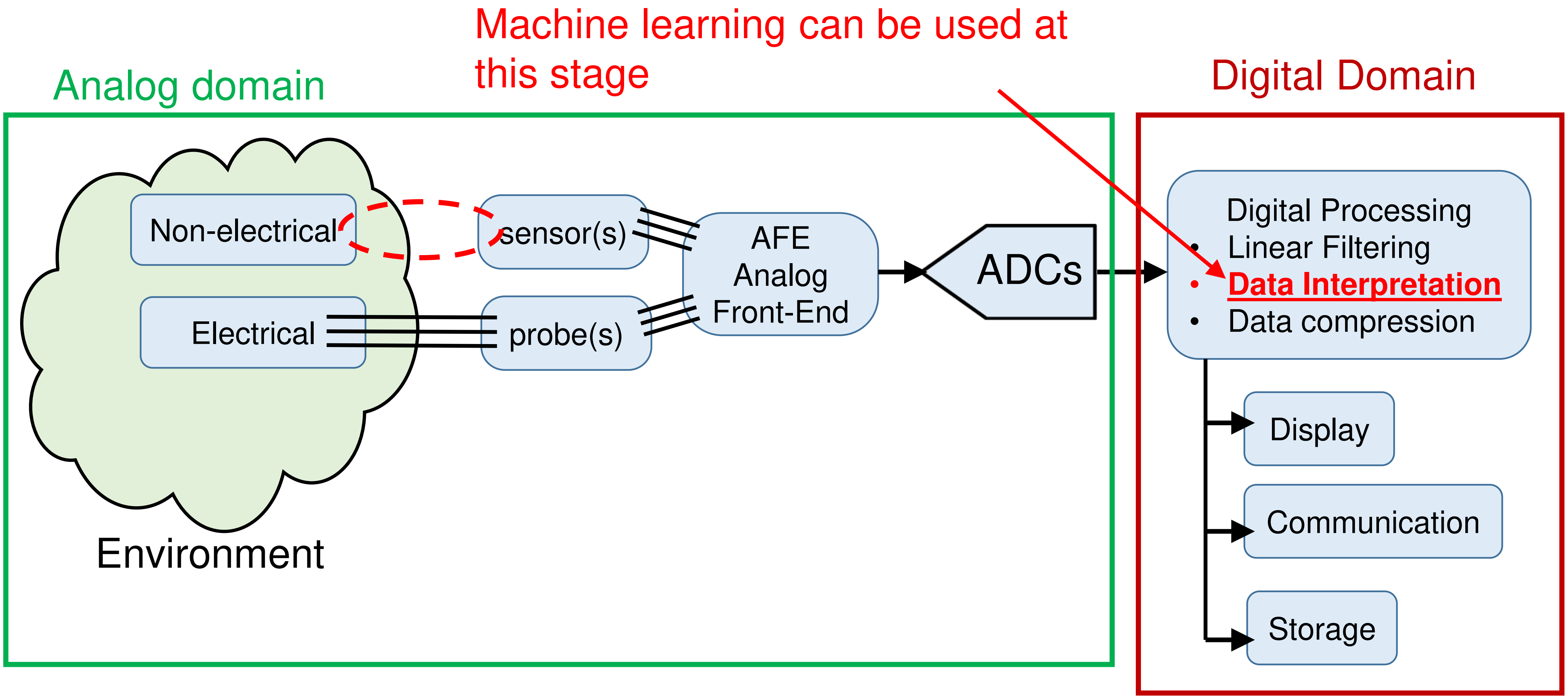
Reinforcement algorithms learn how to perform actions in response to particular situations with the goal to obtain a certain result.

Examples of goals are maximizing the profit of a company, winning a chess match etc ..

The algorithm learns which actions are convenient on the basis of the reward, which is a function of the state. The reward can be positive (win) or negative (loose).

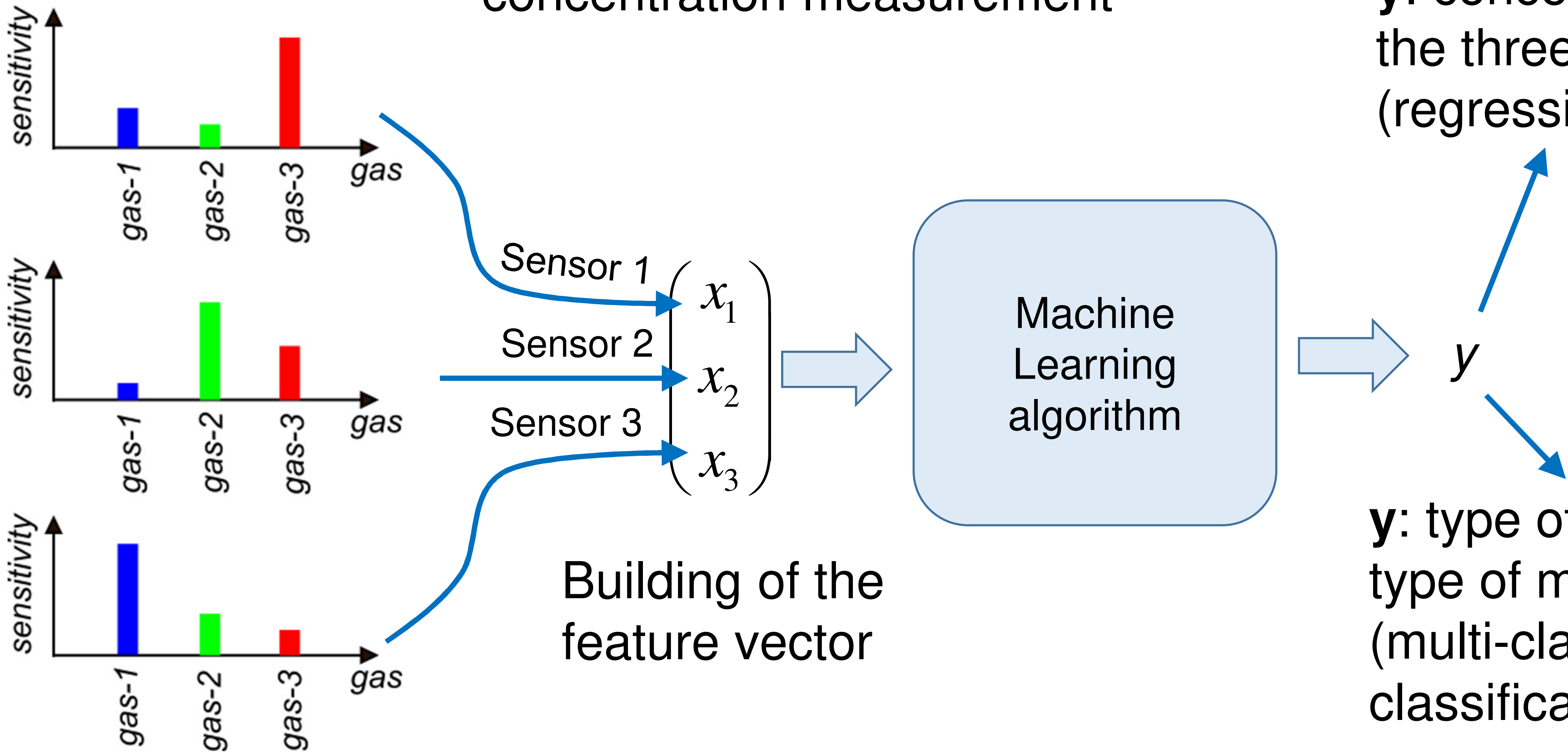


Machine learning in sensor applications



Machine learning in sensor applications

Example: gas classification and/or concentration measurement



Building of the feature vector

y: concentration of the three gases (regression)

y: type of gas or type of mixture (multi-class classification)

When is ML useful for data interpretation in sensors

A priori model tuning
No Machine learning

Data interpretation can be accomplished in two ways:

1. **Model-driven**: we know the deterministic relationship between the quantities that we want to sense and the sensor outputs ("features"). If the inverse relationship (from the features to the quantities) can be easily found, then it is directly implemented in the algorithm.

2. **Data-driven**: if the relationship between from the quantities to the sensor outputs is **<u>difficult-to-model</u>**, due to a large number of features, presence of strong non-linearities, unreliable properties, complicated dependencies, then its better to use a generic model and let the data tune it to perform the required data interpretation.

Machine learning (= statistical learning)

General characteristics of the training process

Hyperparameters: these are not set by the learning process but have to be chosen before the training begins-. Hyperparameters affect:

a) The structure of the model
b) The way the training process is performed

For example, in a polynomial regression process, an hyperparameter can be the <u>degree</u> of the polynomial.

Cost or loss function: measure the difference between the desired outputs and the predicted values

$\hat{y}$:predicted by the model

Example of loss function $$\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2$$

In all phases, proper <u>cost functions</u> are used to assess the degree of accuracy.

Data used in the training process

Training data	Used to really train the algorithm (i.e. set the model parameters)
Development data	Used to choose the hyperparameters
Test data	Used at the end of the learning process to test the accuracy of the model

Effectiveness metrics: the confusion matrix

The confusion matrix is used to represent the accuracy of a classification algorithm

Example with 4 classes A,B,C,D

Predicted labels

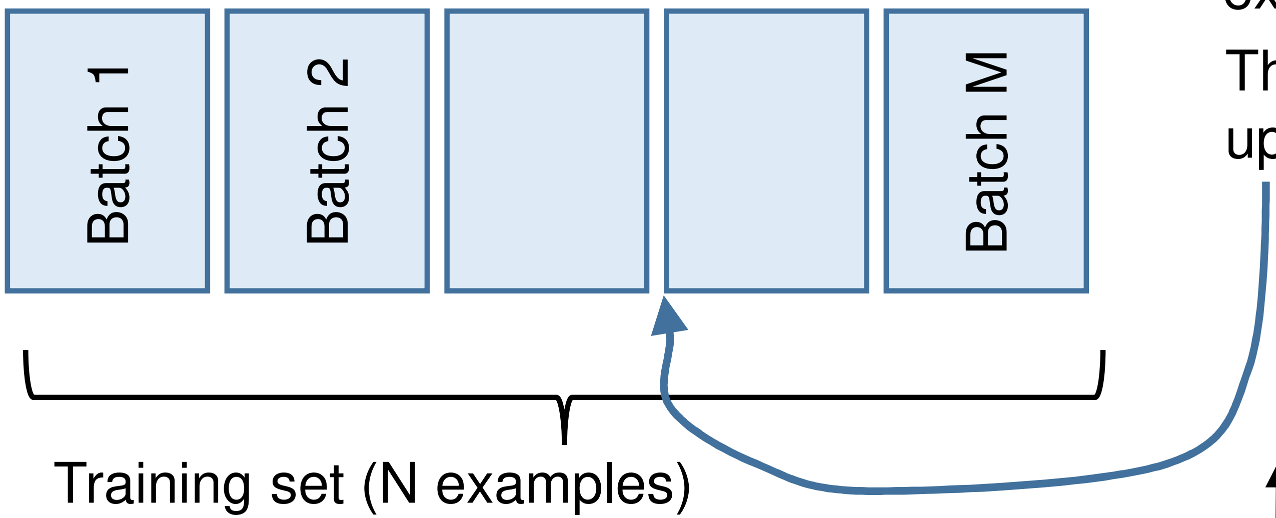
Actual labels	A	B	C	D
A	R_A	W_{AB}	W_{AC}	W_{AD}
B	W_{BA}	R_B	W_{BC}	W_{BD}
C	W_{CA}	W_{CB}	R_C	W_{DC}
D	W_{DA}	W_{DB}	W_{CD}	R_D

Number of examples that have label A and the algorithm classified ad D

In the ideal case, there are non-zero elements only in the diagonal.

Non-zero values in non-diagonal elements represent errors, i.e. "confusion".

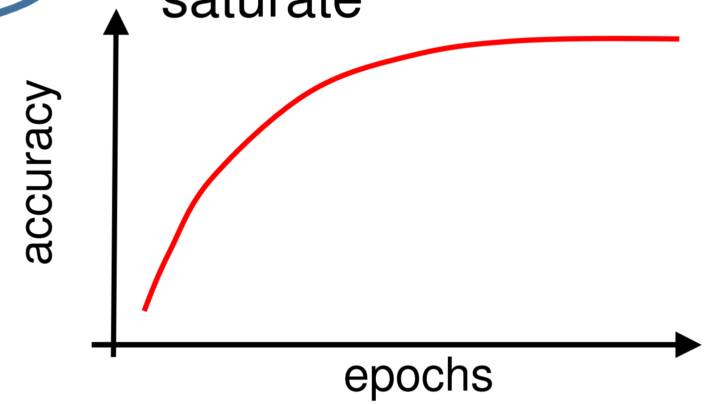
The training phase



Every batch is formed by N / M examples

The model parameters are updated at the end of each batch

The accuracy increases with the number of epochs but tends to saturate



Generally, the same training set is passed through the learning algorithm several times. A single pass of the whole training set is called epoch

Types of ML algorithms

- Decision trees (classification)
- Regression Analysis (regression, continuous target y)
- K-Nearest-Neighbors (k-NN) (classification)
- Logistic Regression (classification)
- Principal Component Analysis (PCA) (clustering)
- Support Vector Machine (SVM): classification

❑ **Artificial Neural Networks (ANN) (classification and /or regression)**

Artificial Neural Networks: the origins

The predecessor: **<u>The Perceptron</u>**
(Ronsenblatt, Cornell Aerounautical Lab.1958)

Originally implemented on a IBM 704 computer and connected to an array of photodetector to obtain a visual classifier.

Inspired by:

1943: First models of the natural neuron (McCulloch-Pitt)

Followed by:
1960: Adaline (ADAptive LInear NEuron)
(B. Widraw et al., Stanford)


Vol. VI, No. 2, Summer 1958

research trends

CORNELL AERONAUTICAL LABORATORY, INC., BUFFALO 21, NEW YORK

The Design of an Intelligent AUTOMATON

by FRANK ROSENBLATT

Introducing the perceptron — A machine which senses, recognizes, remembers, and responds like the human mind.

STORIES about the creation of machines having human qualities have long been a fascinating province in the realm of science fiction. Yet we are now about to witness the birth of such a machine — a machine capable of perceiving, recognizing, and identifying its surroundings without any human training or control.

Development of that machine has stemmed from a search for an understanding of the physical mechanisms which underlie human experience and intelligence. The question of the nature of these processes is at least as ancient as any other question in western science and philosophy, and, indeed, ranks as one of the greatest scientific challenges of our time.

Our understanding of this problem has gone perhaps as far as had the development of physics before Newton. We have some excellent descriptions of the phenomena to be explained, a number of interesting hypotheses, and a little detailed knowledge about events in the nervous system. But we lack agreement on any integrated set of principles by which the functioning of the nervous system can be understood.

We believe now that this ancient problem is about to yield to our theoretical investigation for three reasons:

First, in recent years our knowledge of the functioning of individual cells in the central nervous system has vastly increased.

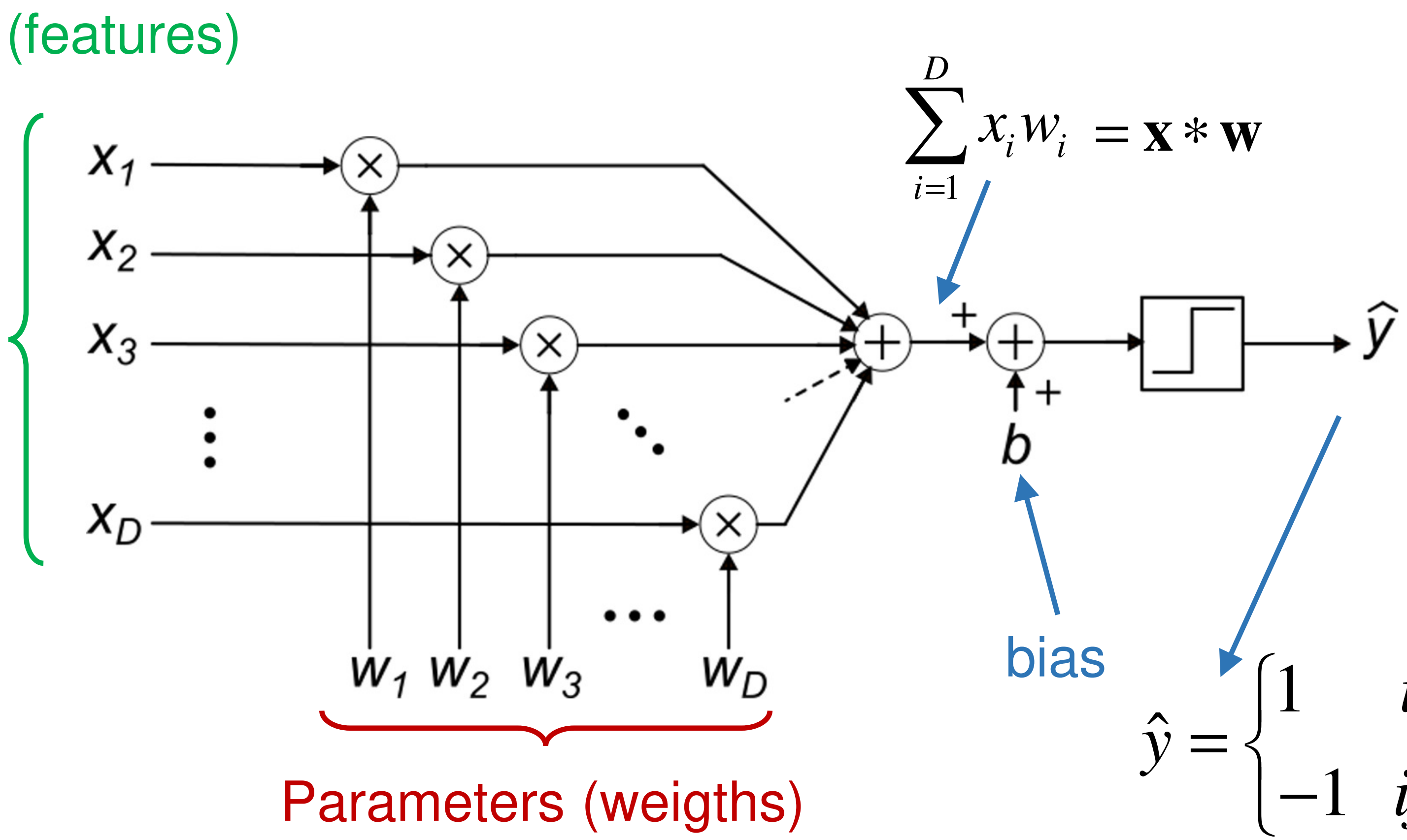
Second, large numbers of engineers and mathematicians are, for the first time, undertaking serious study of the mathematical basis for thinking, perception, and the handling of information by the central nervous system, thus providing the hope that these problems may be within our intellectual grasp.

Third, recent developments in probability theory and in the mathematics of random processes provide new tools for the study of events in the nervous system, where only the gross statistical organization is known and the precise cell-by-cell "wiring diagram" may never be obtained.

Receives Navy Support

In July, 1957, Project PARA (Perceiving and Recognizing Automaton), an internal research program which had been in progress for over a year at Cornell Aeronautical Laboratory, received the support of the Office of Naval Research. The program had been concerned primarily with the application of probability theory to

The perceptron classifier



$$\sum_{i=1}^{D} x_i w_i = \mathbf{x} * \mathbf{w}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix}$$

$$\hat{y} = \begin{cases} 1 & if \quad \mathbf{x} * \mathbf{w} + b \geq 0 \\ -1 & if \quad \mathbf{x} * \mathbf{w} + b < 0 \end{cases}$$

The perceptron operation: 2D case

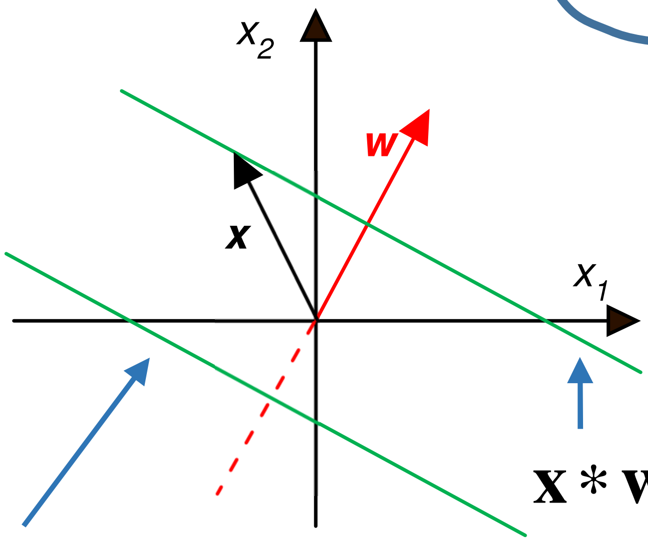
$$\hat{y} = \begin{cases} 1 & if \quad \mathbf{x} * \mathbf{w} + b \geq 0 \\ -1 & if \quad \mathbf{x} * \mathbf{w} + b < 0 \end{cases}$$

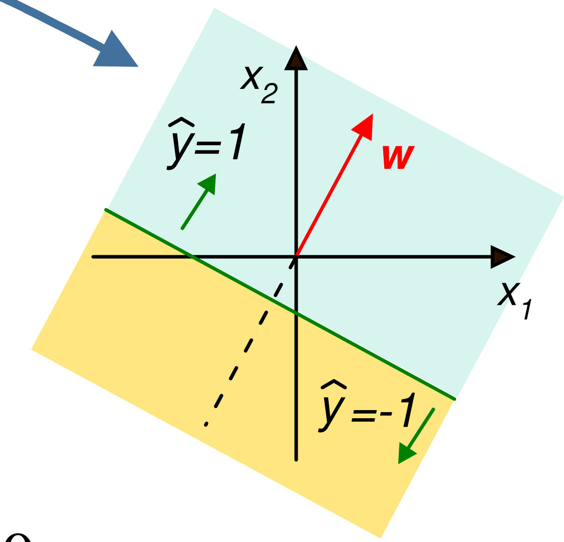
$$\mathbf{x} * \mathbf{w} \geq -b$$

$$\mathbf{x} * \mathbf{w} < -b$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$





$$\mathbf{x} * \mathbf{w} = constant > 0$$

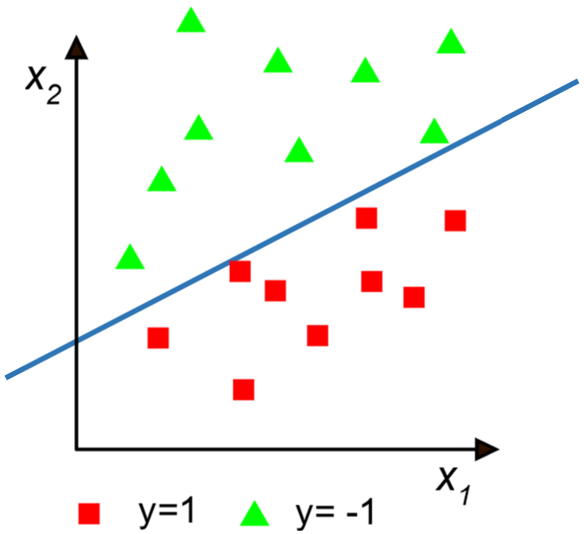
$$\mathbf{x} * \mathbf{w} = constant < 0$$

Perceptron classifier

Labels y (0 or 1)

Training data

$$\begin{pmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(1)} & x_2^{(1)} \\ x_1^{(3)} & x_2^{(3)} \\ \cdots\cdots & \cdots\cdots \\ x_1^{(N)} & x_2^{(N)} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ \cdots \\ 1 \end{pmatrix}$$

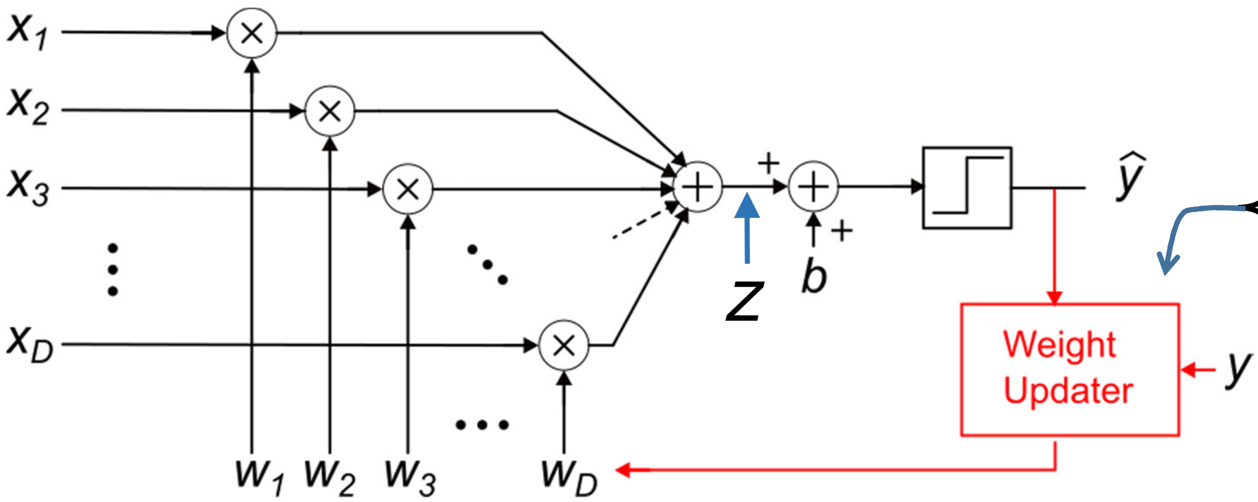


Target: find the correct weight set (w_1, w_2) and bias *b* that allow the perceptron to predict the label on a new data. In the case shown above, the data are "linearly separable", i.e. a line exists that divide the x_1,x_2 plane into two regions where points with homogeneous label are present

Since the plane exists, this classification problem can be solved by a perceptron

The problem is: how to make the perceptron set the weights and the bias autonomously.

Setting the weights from the training data = **learning**



Update algorithm

$$\begin{cases} b(n+1) = b(n) + \eta\left(y_i - \hat{y}_i\right) \\ w_i\left(n+1\right) = w_i(n) + \eta\left(y_i - \hat{y}_i\right)x_i \end{cases}$$

i: 1, ... D

η : learning rate

$$z = \sum_{i=0}^{D} x_i w_i$$

$$w_i\left(n+1\right) = w_i(n) + \eta\left(y_i - \hat{y}_i\right)x_i$$

i: 0, ... D

$$w_0 = b \quad x_0 = 1$$

Artificial Neural Networks

The perceptron implements a "single layer" ANN and can solve only binary classification problems for linearly separable data

An ANN uses more perceptrons (modified to produce a continuous output) to solve a much wider variety of problems