Artificial Intelligence

academic year 2024/2025

Giorgio Fumera, Ambra Demontis

Pattern Recognition and Applications Lab
Department of Electrical and Electronic Engineering
University of Cagliari (Italy)

Introduction to Machine Learning

Outline

- Solving problem by searching
- Solving problems employing knowledge and reasoning
- Solving problems employing uncertain knowledge and reasoning
- **Solving problems learning by examples**

Suggested textbooks

- S. Russell, P. Norvig, *Artificial Intelligence – A Modern Approach*, Pearson, 2003, 2nd ed. (or another edition)
- C.M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995
  - ch. 1 (par. 1.1 to 1.6): Statistical Pattern Recognition
  - ch. 3: Single-Layer Networks
  - ch. 4: The Multi-layer Perceptron
- M. Nielsen, *Neural Networks and Deep Learning*, `http://neuralnetworksanddeeplearning.com/`

Machine learning

There are many tasks, such as recognizing numbers or faces, that humans can easily carry out but don't know **how**.
Therefore, no algorithmic solution is known.
However, **examples** of the desired input-output behaviour ("**training set**") can be easily provided to a machine

Machine learning algorithms learn to perform a task from a set of examples.

Historical notes

- Machine learning was inspired by humans' learning capabilities
- Early ideas (1950s): programming a computer to **learn from experience**, avoiding direct programming (e.g., learning to play draughts by automatic analysis of expert players' games)
- Early applications in **pattern recognition** and **computer vision** tasks, e.g.:
  - optical character recognition (OCR) form **noisy** images, as an alternative to template matching

  - aerial image recognition

Historical notes

- Main tools until the 1980s: **artificial neural networks**, considered at the fringe of AI in those days
- Since the 1990s machine learning becomes a mainstream AI approach to automatically extract useful knowledge from data, thanks to
  - **automated gathering** of large amounts of training data already in **digital form**, from various kinds of sources: the Web (e.g., browsing data, search engines, e-commerce transactions), biological data, social networks (e.g., “likes”), etc.
  - **inexpensive storage**
  - development of methodologies strongly rooted in **statistics**
- Paradigm shift in AI: from **knowledge-driven** to **data-driven** approaches

Main current applications

Machine learning is still widely used nowadays in **pattern recognition** and **computer vision** tasks:

- optical character recognition (printed and handwritten)
- content-based image retrieval (e-commerce product retrieval)
- object detection, localization, recognition, tracking and re-identification in images and videos
- medical image analysis (skin tumor detection)
- biometric identity recognition (fingerprint, face, iris, etc.) `https://www.youtube.com/watch?v=wr4rx0Spihs`
- ...

Main current applications

Many other applications beside computer vision:

- natural language processing (understanding and translation)
- text categorization, topic detection in texts
- speech recognition (transcription) and understanding
- data mining
- computer security, e.g., spam/phishing detection, intrusion detection in computer systems and networks, botnet detection, malware detection
- recommender systems, on-line advertisement (user behaviour modelling, e.g., from social networking data)
- automated (high-frequency) trading
- ...

What is machine learning suitable for?

Machine learning is suited to tasks where **no algorithmic solution is known** (even if humans can easily carry them out), but **examples of the desired input-output relationship** can be provided to a machine. Some application examples:

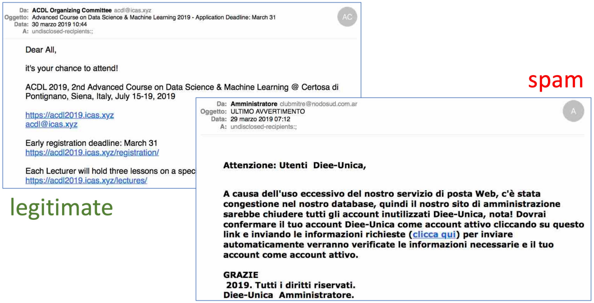
- text categorisation, e.g., spam e-mail recognition
- handwritten digit image recognition
- pedestrian detection in images

Machine learning is **not** suited, instead, to applications where algorithmic solutions are known, e.g., summing any two numbers.

Example: spam e-mail recognition

We can recognise at a glance a spam e-mail in our inbox.

It is also easy to collect from our inbox a set of e-mails, and **manually** label them as “legitimate” or “spam”:



Da: ACDL Organizing Committee acdl@icas.xyz
Oggetto: Advanced Course on Data Science & Machine Learning 2019 - Application Deadline: March 31
Data: 30 marzo 2019 10:44
A: undisclosed-recipients:;

Dear All,

it's your chance to attend!

ACDL 2019, 2nd Advanced Course on Data Science & Machine Learning @ Certosa di Pontignano, Siena, Italy, July 15-19, 2019

https://acdl2019.icas.xyz
acdl@icas.xyz

Early registration deadline: March 31
https://acdl2019.icas.xyz/registration/

Each Lecturer will hold three lessons on a spec
https://acdl2019.icas.xyz/lectures/

Da: Amministratore clubmitre@nodosud.com.ar
Oggetto: ULTIMO AVVERTIMENTO
Data: 29 marzo 2019 07:12
A: undisclosed-recipients:;

Attenzione: Utenti Diee-Unica,

A causa dell'uso eccessivo del nostro servizio di posta Web, c'è stata congestione nel nostro database, quindi il nostro sito di amministrazione sarebbe chiudere tutti gli account inutilizzati Diee-Unica, nota! Dovrai confermare il tuo account Diee-Unica come account attivo cliccando su questo link e inviando le informazioni richieste (clicca qui) per inviare automaticamente verranno verificate le informazioni necessarie e il tuo account come account attivo.

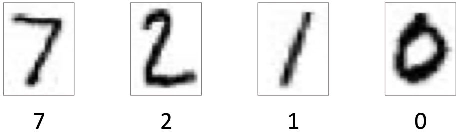
GRAZIE
2019. Tutti i diritti riservati.
Diee-Unica Amministratore.

However, how could we design a spam filter, i.e., an algorithm that **automatically** labels incoming e-mails as “legitimate” or “spam”?

Example: handwritten digit recognition

We all can recognise (more or less easily) a large variety of handwritten characters.

Also in this case it is easy to collect a large sample of handwritten text (e.g., from different writers), and then **manually** segmenting and labelling the individual character images:

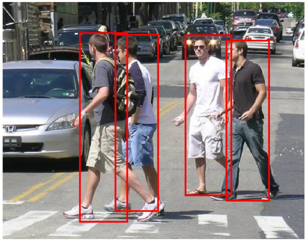


However, how could we design an algorithm that **automatically** recognises handwritten character images?

Example: pedestrian detection in images

We can easily detect the presence of pedestrians in our field of view, e.g., when driving a car.

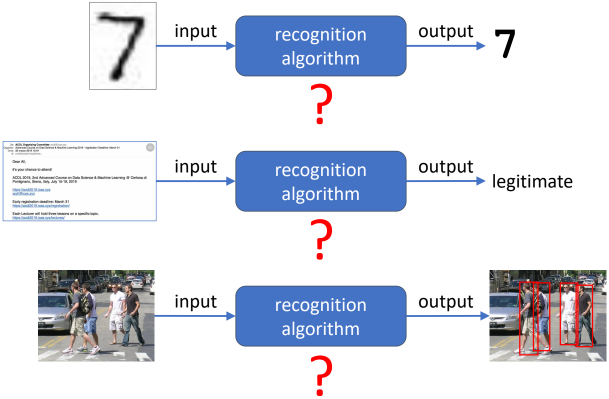
It is also easy to collect a set of images, e.g., from a camera mounted on a car, and to **manually** draw a tight *bounding box* around each pedestrian (if any).

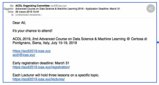


However, how could we design an algorithm that **automatically** detects pedestrians on images acquired by a video camera, e.g., mounted on a self-driving car?

The machine learning approach

If no algorithmic solution is known, it is infeasible to **explicitly** design a recognition algorithm:




Da: ACDL Organizing Committee acdl@icas.xyz
Oggetto: Advanced Course on Data Science & Machine Learning 2019 - Application Deadline: March 31
Data: 30 marzo 2019 10:44
A: undisclosed-recipients:;

Dear All,

it's your chance to attend!

ACDL 2019, 2nd Advanced Course on Data Science & Machine Learning @ Certosa di Pontignano, Siena, Italy, July 15-19, 2019

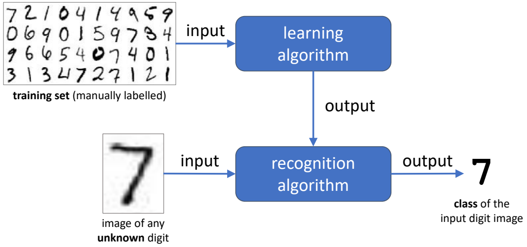
https://acdl2019.icas.xyz
acdl@icas.xyz

Early registration deadline: March 31
https://acdl2019.icas.xyz/registration/

Each Lecturer will hold three lessons on a specific topic.
https://acdl2019.icas.xyz/lectures/

The machine learning approach

Nevertheless, the availability of examples of the desired input-output behaviour suggests a different approach: devising a **learning** algorithm that **automatically** builds a recognition algorithm, based on the available examples (training set):

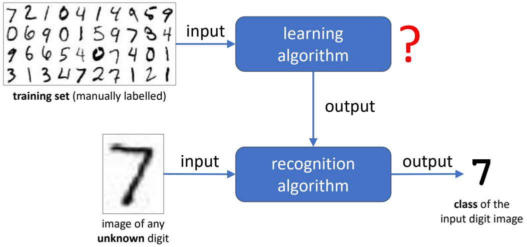


input
learning algorithm
training set (manually labelled)
output
input
recognition algorithm
output
7
image of any unknown digit
class of the input digit image

This approach is known as **learning from examples**, or **supervised learning**, where the "supervision" consists in the examples of the desired input-output behaviour.

The machine learning approach

It may seem that the problem has just been shifted to a higher level: instead of directly designing a **recognition** algorithm, one should design a **learning** algorithm... But, **how** can one do that?



training set (manually labelled)
input
learning algorithm
?
output
image of any unknown digit
input
recognition algorithm
output
7
class of the input digit image

However, we shall see that general (i.e., not application-specific) and effective solutions exist.

Main machine learning paradigms

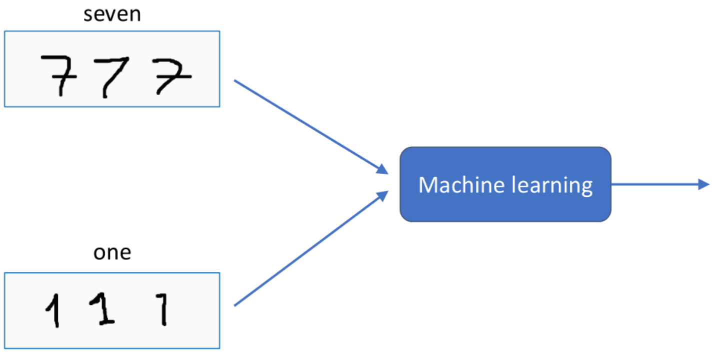
Supervised learning is not the only type of problem in the machine learning field.

The main machine learning paradigms can be categorised into:

- **supervised** learning
- **semi-supervised** learning
- **unsupervised** learning
- **reinforcement** learning

Supervised learning

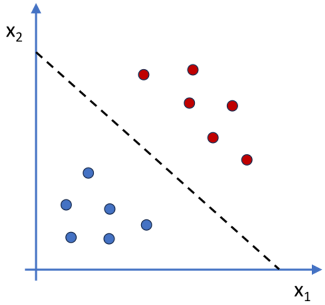
Examples of the desired input-output behaviour are available.



Supervised learning

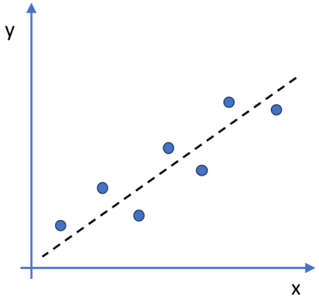
Classification: predicting the **class** (or category) of input "objects", out of a predefined set of classes, e.g.:

- e-mail spam filtering: two-class problem (spam, legitimate)
- handwritten digit recognition: ten-class problem (0, 1, . . . , 9)



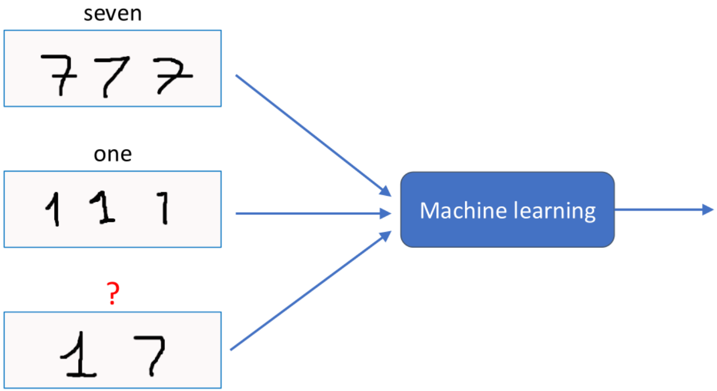
Supervised learning

Regression: predicting a **numerical value**, e.g., crowd counting: how many people are there in a given video frame?



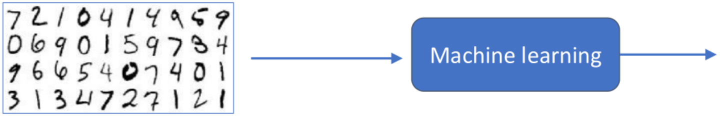
Semi-supervised learning

Some examples of the desired input-output relationship are available.



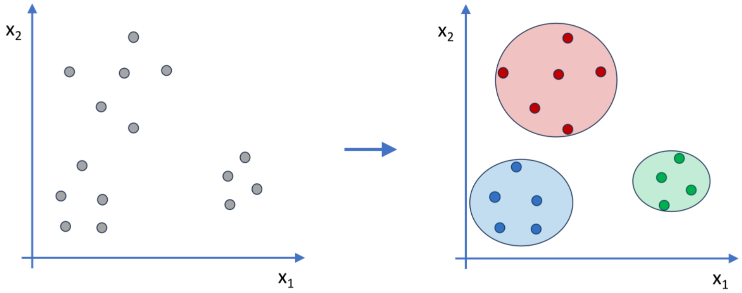
Unsupervised learning

No examples of the desired input-output relationship are available.



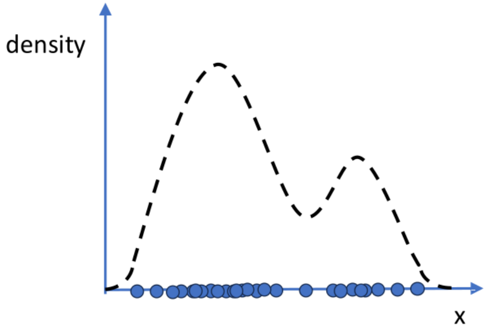
Unsupervised learning

Clustering: discovering groups of similar examples, e.g., different *malware* families



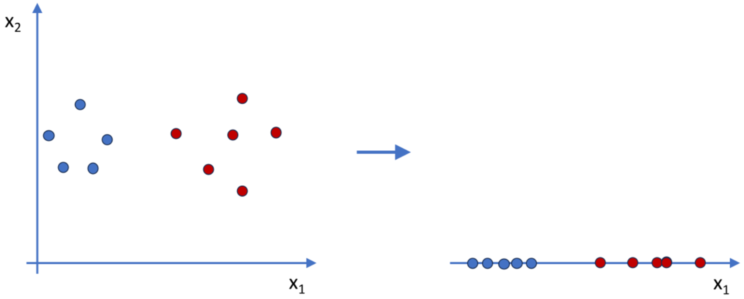
Unsupervised learning

Density estimation: determining the probability distribution of data from a finite set of samples



Unsupervised learning

Visualisation: projecting high-dimensional data in a number of dimensions that can be shown in a plot (three or fewer), preserving their “structure”

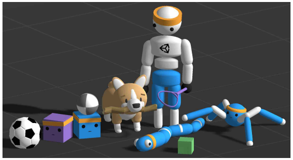


Reinforcement learning

Finding an optimal **action policy** (i.e., the one that maximises a given "reward"), based on examples of the outcome of **entire courses of action**, e.g., either success or failure (no information is available on **single** actions).

Application examples:

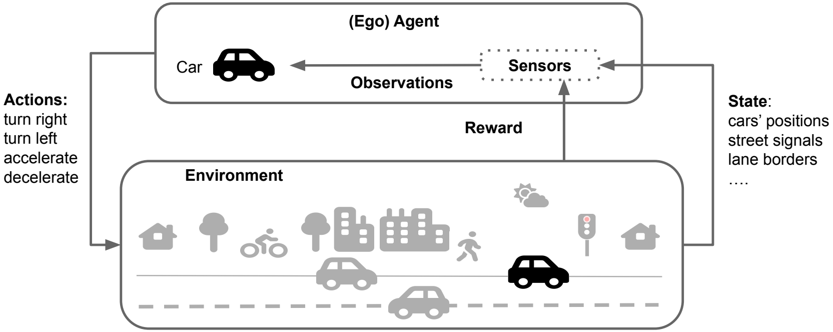
- **robot control**
- **game playing**: backgammon, non-playing characters in video games, etc.



https://github.com/Unity-Technologies/ml-agents

Reinforcement learning

The agent learns from a series of reinforcements: rewards and punishments received for its actions by the environment.



Supervised classification

This course focuses on supervised classification problems, which are very common in real-world applications, e.g.:

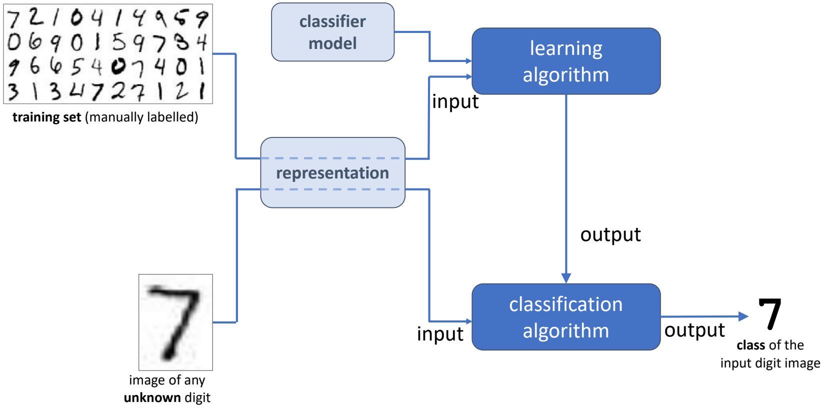
- optical character recognition (OCR)
- biometric identity recognition (e.g., face and fingerprints)
- text categorisation (e.g., news tagging, e-mail spam filtering)
- scene recognition from images (e.g., indoor, outdoor, urban)
- computer security (e.g., *malware* detection)
- speech recognition (or speech-to-text): translating spoken language into text (does not include language **understanding**)
- ...

In the following we shall first see how to **formulate** supervised classification problems and their main **issues**, then two well-known supervised classifiers: **decision trees** and **artificial neural networks**.

E-mail spam filtering and **handwriting digit recognition** will be used as application examples.

Supervised classification: problem formulation

A more complete scheme of a supervised classifier design process, using handwritten digit recognition as an example: each component is discussed in detail in the next slides.



Supervised classification: problem formulation

Main elements of the *classifier design process*:

- defining the classification task:
  - what are the **"objects"** to be classified? (e.g., images, e-mails)
  - what are the **classes**?
    (e.g., the ten digits 0, 1, . . . , 9; spam and legitimate e-mails)
- collecting and labelling (usually, manually) a **training set**
- choosing a **representation** for the "objects" to be classified
- choosing a **hypothesis space** (also known as **classifier model** in the case of supervised classification problems)
- choosing a **learning algorithm**

Defining the classes

In standard classification problems the set of possible classes is **finite**, and has to be defined beforehand.

Classes can be represented for convenience in different ways, e.g., using symbols or numbers, which are usually called **class labels**.

The set of class labels, i.e., the possible values of the **output** of the classification algorithm, will be denoted in the following by $\mathcal{Y}$.

Examples of a possible choice of class labels:

- e-mail spam filtering: $\mathcal{Y} = \{\texttt{Spam}, \texttt{Legitimate}\}$
- handwritten digit recognition: $\mathcal{Y} = \{0, 1, \ldots, 9\}$

Collecting a training set

Intuitively, the **training set** should be as much **representative** as possible of the classes of interest; for instance:

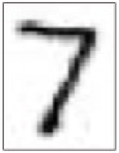
- for **e-mail spam filtering**, it should include examples of the different kinds of spam and legitimate e-mails received by a specific user (e.g., work and personal legitimate e-mails)
- for **handwritten digit recognition**, it should include samples from different writers or handwriting styles

Usually, to enlarge representativeness it is necessary to increase the **size** of the training set. However, in supervised learning problems a larger training set requires a higher **manual annotation effort**: a trade-off between training set size and representativeness must therefore be reached.

Choosing a representation for the "objects" to be classified

A suitable representation of the raw "objects" to be classified should be chosen, as the input of the learning and classification algorithms. The space of all possible object representations, i.e., the possible **input** values of the classification algorithm, will be denoted by $\mathcal{X}$.

For instance, if the raw data for a **handwritten digit recognition** task consists in grey-level images with size 28×28 pixels and 8 bits per pixel, $\mathcal{X}$ is the set of all possible 28×28 matrices whose elements are integers in the range $[0, 255]$.



However, this representation is likely to be **redundant**, which does not help the task of the classifier.

Similarly, for **spam e-mail filtering** the original representation of e-mails according to existing standards (e.g., MIME) may not be the "best" one for a classifier.

Choosing a representation for the "objects" to be classified

A widely used alternative to the the original, raw data representation is to extract from the "objects" to be classified a smaller set of measures, called **attributes** in machine learning and **features** in pattern recognition, that retain sufficient information to discriminate between the different classes of interest, i.e., that exhibit **discriminant capability**.

Each "object" can then be represented as a fixed-size **attribute/feature vector** $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is called **attribute/feature space**. For instance:

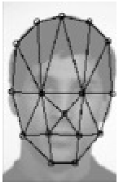
- **handwritten digit recognition**: suitable attributes could be obtained from the orientation of edge pixels
- **spam e-mail filtering**: analogously to more general text categorisation problems, a typical choice is to focus on the textual content of a message (subject and body), e.g., checking whether each of a predefined set of terms $\{\mathrm{t}_1, \ldots, \mathrm{t}_d\}$, that are more likely to occur either in spam or in legitimate e-mails, is present, which can be represented as a vector of d Boolean values: $\mathcal{X} = \{\mathrm{True}, \mathrm{False}\}^d$

Choosing a representation for the “objects” to be classified

Several kinds of representations have been defined in the machine learning and pattern recognition fields; for instance:

- **fixed-size attribute (feature) vectors**, e.g.:
  - OCR: numerical values encoding edge pixel orientation
  - text categorisation: Boolean values encoding the occurrence of a predefined set of category-related terms
- **strings** or **graphs**, e.g.:
  - face recognition: graph of *fiducial points* (e.g., pupils, mouth corners, nose tip)

Wiskott et al., *Face Recognition by Elastic Bunch Graph Matching*, IEEE T-PAMI, 1997



- **sentences in a logical language**, e.g.:
  - predicate logic representation of the position of pieces and of the moves in chess games, to learn strategy rules

Choosing a representation for the "objects" to be classified

Formally, given a representation space $\mathcal{X}$ and a set of class labels $\mathcal{Y}$, every "object" can be seen as a pair $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$.

A classification algorithm can therefore be seen as a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that associates to a given input $\mathbf{x} \in \mathcal{X}$ (e.g., a vector of d Boolean values representing the content of an e-mail)
a **predicted** class label $y \in \mathcal{Y}$ (e.g., either `Spam` or `Legitimate`).

Choosing the *hypothesis space* (classifier *model*)

Classification algorithms usually are not expressed as computer programs, but in many different ways, depending also on the representation of the objects to be classified. Some examples:

- **`IF...THEN` rules**, where the condition part (`IF...`) refers to attribute values and the consequent part (`THEN...`) to one of the classes (e.g., *decision trees*)
- **mathematical functions**, for problems with numerical attributes, with a numerical encoding of the classes (e.g., *artificial neural networks*)
- **logical sentences**, when the objects to be classified are represented by logical sentences themselves

Choosing the *hypothesis space* (classifier *model*)

Usually machine learning methods require, during design, the choice of a **hypothesis space**, which in supervised classification is also called **classifier model**.

The classifier model is the set $\mathcal{H}$ of the possible classification algorithms, among which the learning algorithm will select one.

For instance, *decision trees* and *artificial neural networks* are examples of classifier models.

Classifier model: an example

Consider a **spam e-mail filtering** problem, where:

- e-mails are represented by **feature vectors** denoting the occurrence of a predefined set T of d terms, $\mathbf{x} = (x_1, \ldots, x_d) \in \mathcal{X} = \{0, +1\}^d$, where the values 0 and $+1$ denote the absence and the presence of a term, respectively
- class labels: $\mathcal{Y} = \{\texttt{Spam}, \texttt{Legitimate}\}$

A simple classification algorithm is defined through a linear function

$$f(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{d} w_i x_i - w_0 \ ,$$

whose coefficients $\mathbf{w} = (w_0, w_1, \ldots, w_d) \in \mathbb{R}^{d+1}$ have to be set by a learning algorithm.

Classifier model: an example

For given coefficients **w** the classifier is defined as:

$$h(\mathbf{x};\mathbf{w}) = \begin{cases} \texttt{Spam}, & \text{if } f(\mathbf{x};\mathbf{w}) \geq 0 \\ \texttt{Legitimate}, & \text{otherwise} \end{cases}$$

In words, the rationale of this kind of classifier is the following: if the sum of the weights of the terms in T that appear in an e-mail is negative, that e-mail is predicted to be legitimate, otherwise it is predicted to be spam.

Intuitively, the learning algorithm will assign **positive** weights to terms more likely to appear in spam e-mails, and **negative** weights to terms more likely to appear in legitimate ones.

The corresponding **classifier model** is therefore the set of all possible functions $h(\cdot;\mathbf{w})$:

$$\mathcal{H} = \{h(\cdot;\mathbf{w}) : \mathcal{X} \mapsto \mathcal{Y} \mid \mathbf{w} \in \mathbb{R}^{d+1}\}$$

Choosing a learning algorithm

Once the **classifier model** $\mathcal{H}$ has been chosen, a **learning algorithm** suitable to it has to be chosen.

The goal of the learning algorithm is to select one classifier $h \in \mathcal{H}$, taking into account the examples of the desired input–output behaviour in the **training set**.

Several learning algorithms have been defined so far, for each possible classifier model (e.g., for decision trees and for artificial neural networks): the task of the **designer** amounts therefore to choose one of the available learning algorithms for the selected classifier model.

Supervised learning: generalisation capability

The goal of supervised learning is to find a “good” predictor for **all possible** instances, including **unseen** ones, i.e., instances not present in the training set: this is called **generalisation capability**.

For classification problems, this means that a classifier should be capable of correctly **predicting** the class of **any** instance, e.g.:

- recognising handwritten digits by different writers, or in different writing styles, possibly not included in the training set
- recognising spam and legitimate e-mails that will be received by a given user **after** the deployment of a spam filter

However, how can a good generalisation capability be achieved, based on a finite set of examples of the desired input-output behaviour?

Supervised learning: generalisation capability

Learning from examples is a form of a general procedure known in logic as **induction**: the inference of a **general** law from **particular** instances.

Induction is widely used to formulate **scientific theories** in empirical disciplines like physics, and is widely studied in the **philosophy of science**.

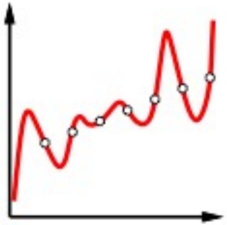
Inductive learning problems are however known to be **ill-posed**, since many different explanations may agree with the observations (examples) at hand: how to find the “correct” one?

Supervised learning: generalisation capability

A simple example of an induction procedure is represented by **interpolation** problems, which can occur, e.g., in **physics**, when trying to formulate a law representing the relationship between two quantities x and y based on a finite set of **noisy** observations $\{(x_1, y_1), \ldots (x_n, y_n)\}$.

For instance, **infinitely** many polynomials of degree n or higher can interpolate **with no error** any set of n points:

- which is the "correct" one?
- what if the "true", unknown physical law cannot be represented by a polynomial?



Supervised learning: generalisation capability

In supervised learning several solutions have been devised for the inductive learning problem of finding a hypothesis with a good generalisation capability based on a **finite** set of examples.

Existing solutions are all based on two general induction principles:

- **consistency** with the observations: the hypothesis should **agree** with the available examples
- **minimal complexity**: the hypothesis should be as **simple** as possible – the rationale is that a simple hypothesis which agrees with the observations is more likely to be correct than a complex one (a principle known as **Occam's razor**, from the 14th century logician William of Occam)

Supervised learning: generalisation capability

For supervised classification the two principles above can be restated as follows: given a hypothesis space $\mathcal{H} = \{h : \mathcal{X} \mapsto \mathcal{Y}\}$ and a training set $\mathcal{T}$, the "best" hypothesis (classifier) in $\mathcal{H}$ is the "**simplest**" classifier among the ones **consistent** with $\mathcal{T}$.

Formally, a hypothesis h is consistent with $\mathcal{T}$, if it outputs the correct class label for **all** the examples in $\mathcal{T}$:

$$h(\mathbf{x}_i) = y_i, \quad \text{for every } (\mathbf{x}_i, y_i) \in \mathcal{T} \ .$$

In practice, the **consistency** and **minimal complexity** principles can be implemented in different ways for different classifier models and different learning algorithms: in this course possible solutions for *decision trees* and *artificial neural networks* will be presented.

Achieving generalisation capability: some issues

Several issues arise when trying to implement the **consistency** and **minimal complexity** principles:

- how to **evaluate** the "complexity" of a hypothesis (classifier)?
- what is the **computational complexity** of finding the "simplest" consistent hypothesis?
- what if **no** consistent hypotesis exists in $\mathcal{H}$?
- in real-world problems the "correct" hypothesis is **unknown**: what if it does **not** belong to $\mathcal{H}$?
- what if the feature values come from **noisy** measurements? In this case enforcing consistency may be counterproductive...

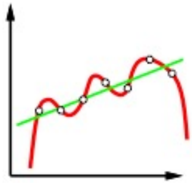
Another issue is how to **evaluate** generalisation capability itself.

Generalisation capability: the *over-fitting* issue

A typical issue of the inductive learning approach is that achieving **consistency** may require a **complex** hypothesis space.
This may lead to **over-fitting**: correctly classifying all training instances, but incorrectly classifying of many unseen ones, i.e., **poor generalisation capability**.

The over-fitting phenomenon is analogous to:

- **exactly** interpolating a set of n observations using a $n-1$ degree polynomial, instead of using a simpler one at the expense of some approximations
- students preparing for exams by rote learning (exact knowledge of the answer to known questions, no knowledge about any other question)



How to evaluate generalisation capability?

Evaluating generalisation capability is an issue in itself:

- what **measure** to use? (e.g., is the fraction of misclassified instances a suitable measure?)
- it refers to **all** possible instances, including **unseen** ones: how can it be evaluated during classifier design, given that by definition only training examples are available?

Several solutions have been devised in the machine learning field: the main ones will be presented in this course.

Decision Trees and Artificial Neural Networks

In the following, the above general concepts of supervised classification will be made concrete by showing their application to two well-known classifier models:

- **Decision Trees**
- **Artificial Neural Networks**

They have an historical relevance, as they are among the early classifier models introduced in the Artificial Intelligence field, and are also still widely used today, especially in the form of **Decision Forests** and **Deep Neural Networks**, respectively.

Decision Trees

- Introduced in the 1950s in **psychology** as models of **high-level** human learning (verbal and concept learning)
- Represent `IF...THEN...` classification rules structured as a **tree** graph
- Originally devised for categorical (e.g., Boolean) attributes
- Still widely used today in several applications (e.g., computer vision, bioinformatics), especially as **Random Forests**
- Also called **Classification Trees** to distinguish them from **Regression Trees** used in regression

Artificial Neural Networks

- Introduced in the 1950s as mathematical models of **low-level** human brain functions (neurons, networks of neurons)
- Require numerical attributes
- Used in practical applications since the late 1980s
- Current evolution: **Deep Neural Networks**, and in particular **convolutional neural networks**, mainly used in computer vision and natural language processing