3.1 Introduction to Machine Learning and Potential Applications to Chemical Engineering

Massimiliano M. Villone
University of Naples Federico II, Italy
massimiliano.villone@unina.it

GRICU PhD School 2021

Digitalization Tools for the Chemical and Process Industries

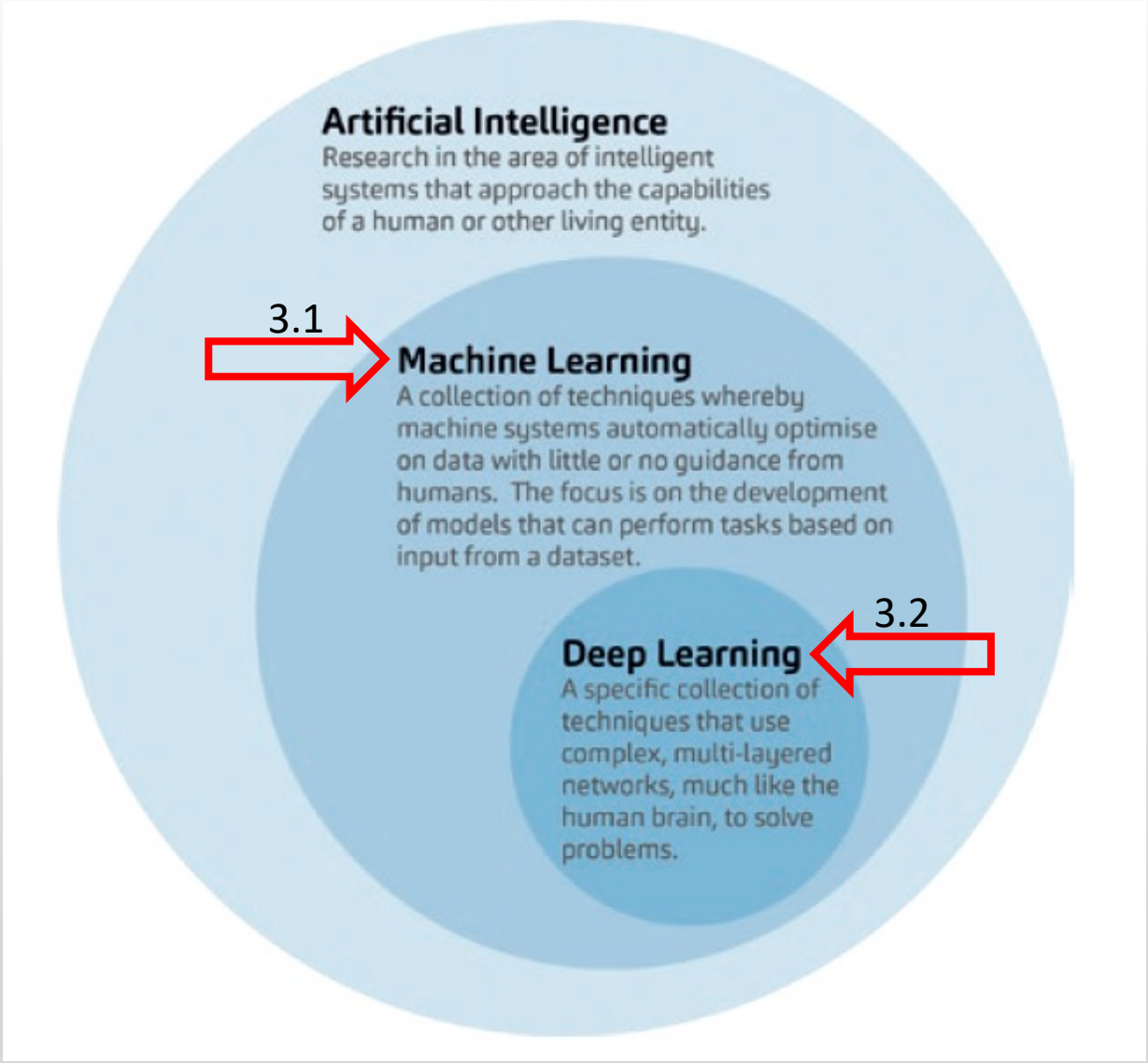
March 18, 2021

Outline

1. Introduction to Machine Learning
2. Classification
3. Regression
4. An application to chemical engineering

What is Machine Learning?

- The term "**Artificial Intelligence**" (**AI**) is colloquially used to describe the "ability" of machines to mimic cognitive functions of human mind, such as **learning** and **problem solving**.
- **Machine Learning** (**ML**) is the study of computer algorithms that improve automatically through experience.
  - Subfield of AI.
  - ML algorithms build models based on data (training examples) in order to make predictions or take decisions.
  - Several (and progressively increasing) **application fields**, e.g.:
    - communication, cyber-security, economics and finance, gaming, linguistics, **manufacturing**, marketing, medicine, robotics, satellite navigation, …
- **Deep Learning** (**DL**) is an important subfield of ML based on **Artificial Neural Networks** mimicking human brain operation (see 3.2).



Gertz F., Fleutsch G., Applications of Deep Learning in Medical Device Manufacturing. *ONdrugDelivery*, 110, 6-11, 2020.

A bit of history

- The term “Machine Learning” was used for the first time in 1959 by **Arthur Lee Samuel**[1].
  - Samuel had an electrical engineering degree from MIT.
  - In 1959, he used to work at IBM when he developed his **checkers program** on IBM701, which is considered the **world's first self-learning program**.
  - He was also among the developers of the TeX typesetting language.

- In 1997, Tom M. Mitchell gave an **operational definition** of ML[2]:
  - “A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance at tasks in *T*, as measured by *P*, improves with experience *E.*”
- Contemporary ML has **two possible objectives**:
  - **classification** of data based on models,
  - **prediction** for future outcomes based on data.

Introduction to ML 1

- To solve a problem on a computer, we typically need an **algorithm** → finite sequence of instructions that transform inputs into outputs.
  - For example, an algorithm for sorting numbers: the input is a set of numbers in random order, the output is the ordered list.
- For the same task, there might be different algorithms.
  - The choice is based on some optimization criteria: the one requiring the less instructions, memory, …
- Sometimes, the inputs and outputs are clearly defined, but there is **no algorithm**.
  - Example: sorting spam email from legitimate one.
    - Input: email messages, output: yes/no
    - How to transform input into output? This may change from person to person…
- When an algorithm is lacking, the problem can be solved anyways if we have **a lot of data**.
  - We feed a machine with thousands of messages, of which we know whether they are spam or not (**training data**), and we ask it to extract "automatically" the algorithm to classify them.

Introduction to ML 2

- ML is programming computers to optimise a performance criterion using example data.
- Given a model containing some parameters, learning is optimising the parameters of the model using the training data in order to
  - enhance the description of the data (descriptive model),
  - predict future data (predictive model).
- Since the core task is making inference from a sample, ML uses the **theory of statistics**.

ML approaches

- ML approaches are traditionally divided into **three categories**:
  - **supervised learning**: a "teacher" feeds the computer with examples of inputs and corresponding "correct" outputs with the aim of learning a general rule that maps inputs to outputs,
  - **unsupervised learning**: unlabelled inputs are given to a learning algorithm, leaving it "on its own" to find a structure among them,
  - **reinforcement learning**: in case the output of a system is a sequence of actions, the single action is not important. Instead, it is important to identify a correct sequence to reach a certain goal, also interacting with a **dynamic environment**.
    - Examples: game playing, driving vehicles, …
  - Intermediate approaches exist, like **semi-supervised learning**, where some of the training examples are unlabelled and some are labelled. The presence of the latter can produce a considerable improvement in learning accuracy.

Applications of ML 1

- Given many data, one application of ML is of finding **association rules** among them.
  - Association rules are **conditional probabilities** of the form $P(Y|X)$.
  - Example: from the data available to a supermarket chain, it emerges that 70 percent of customers who buy beer also buy chips → $P(\text{chips}|\text{beer}) = 0.7$.
- Another typical application is **classification** of inputs based on their features.
  - Classification rules are conditional structures based on **training with data**.
  - Example: in order to loan money, customers of a bank are classified according to the rule $IF\ [(\text{income} > X)\ AND\ (\text{savings} > Y)]\ THEN\ \text{low} - \text{risk}\ ELSE\ \text{high} - \text{risk}$.
  - Classification, in turn, is the basis for **prediction**: if the near future is similar to the past, a rule that fits the available data allows to make correct predictions for novel instances.
  - A more complex example of classification is **pattern recognition**, where there are multiple classes.
    - For example, character recognition, language translation, face recognition, medical diagnosis, ...

Applications of ML 2

- The data, in turn, can be used to learn **rules** from them → **knowledge extraction**.
  - Rules are models explaining the data.
  - This allows for **memory saving**.
  - **Outlier detection**: finding data that do not obey the rule.
- Another important ML application is **regression**.
  - Given input data X, output data Y, and a model $y = g(x, \theta)$, where θ denotes parameters, the ML algorithm optimizes the value of θ such that the approximation error is minimised, i.e., estimates are as close as possible to Y-values given in the training set.
  - Example: coffee consumers' satisfaction is measured at varying temperature, extraction time, and coffee bean type. To find the optimal setting, a regression model linking inputs to output is fitted. Then, new data are collected close to the optimum of the model and a new model is fitted to identify an even better configuration.

Applications of ML 3

- One aim of unsupervised learning is the identification of patterns in large data sets → **data mining**.
- In statistics, the evaluation of pattern recurrence is called **density estimation**.
- A method for density estimation is **clustering**, i.e., finding groupings of inputs.
  - **Example**: document clustering depending on the number of shared words.
    - Critical aspect: lexicon choice.

So, what is ML in the end?

- Given the successful application of ML in various domains, it may be claimed that what we needed was **not new algorithms**[3], but
  - a lot of data,
  - sufficient computational power to run algorithms on them.
- For example, support vector machines are based on **potential functions**, **linear classifiers**, and **neighbour-based methods** proposed in the 1950s and 1960s.
  - No fast computers or large storage devices were available for these algorithms to show their full potential at that time.
- Complicated tasks, e.g., machine translation, can be solved with relatively **simple learning algorithms**, but trained on large amounts of example data.
  - Recent successes with DL algorithms support this claim.
- AI seems not to originate from some outlandish formula, but rather from the patient, almost brute-force use of simple, straightforward algorithms[3].

ML for classification

- On of the main tasks of ML is **classification**.
- Let us go through an **example**: we want a machine decide if a car is a family car or not.
- We show a **set of examples** of cars to people, asking them to **label** them depending on whether they are family cars or not.
  - Yes → positive example
  - No → negative example
- This has two possible aims:
  - **prediction**: given a car not labelled before, is this a family car?
  - **knowledge extraction**: what do people expect form family cars?

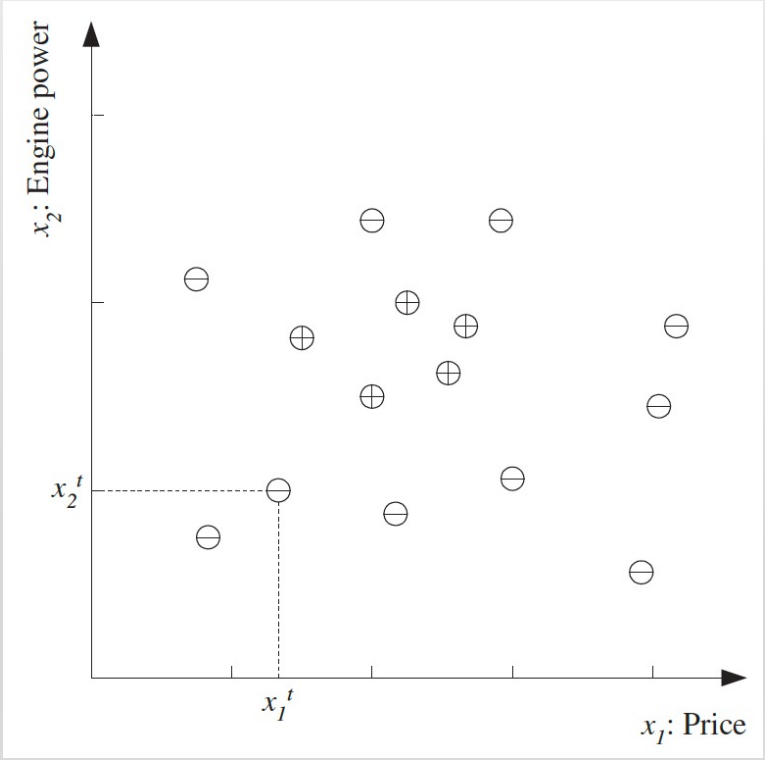
Training examples

- In order to label the examples, the **features** that identify the **class** must be selected, e.g.
  - price x_1,
  - engine power x_2.
- The **training dataset** contains N examples

$$\mathcal{X} = \{\boldsymbol{x}_t, r_t\}_{t=1}^{N}$$

where

- $\boldsymbol{x}_t = [x_1, x_2]$
- $r_t = \begin{cases} 1 \text{ if } \boldsymbol{x}_t \text{ is a positive example} \\ 0 \text{ if } \boldsymbol{x}_t \text{ is a negative example} \end{cases}$

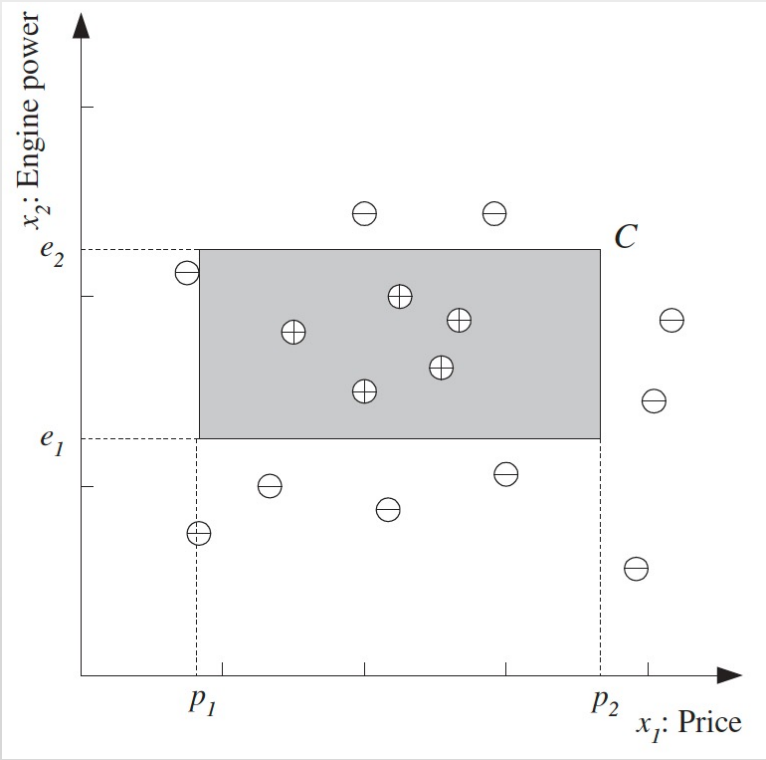


Hypothesis class selection

- After discussion with **"experts"**, we might state that a family car satisfies

$$(p_1 \leq x_1 \leq p_2) \cap (e_1 \leq x_2 \leq e_2)$$

- C is assumed (by the experts) to be a rectangle (with unknown boundaries) in the feature space.
  - C is drawn from an **hypothesis class** $\mathcal{H}$.
  - $\mathcal{H}$ is the class of all the rectangles in the feature space.
- The objective of the learning algorithm is to find $h \in \mathcal{H}$, specified by $\left(p_1^h, p_2^h, e_1^h, e_2^h\right)$, that approximates C as closely as possible.



Hypothesis selection 1

- The **hypothesis** h makes a prediction for an instance $\boldsymbol{x}$ such that

$$h(\boldsymbol{x}) = \begin{cases} 1 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as a positive example} \\ 0 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as a negative example} \end{cases}$$

- In “real life”, we do not know $C(\boldsymbol{x})$, so we cannot evaluate how well $h(\boldsymbol{x})$ matches $C(\boldsymbol{x})$.

- What we have is the training set $\mathcal{X}$, which is a small subset of all possible data.

Hypothesis selection 2

- The **empirical error** is the portion of training instances where predictions of h do not match the required values given in $\mathcal{X}$.
- It is defined as

$$E(h|\mathcal{X}) = \sum_{t=1}^{N} 1(h(\boldsymbol{x}_t) \neq r_t)$$

with $1(a \neq b) = \begin{cases} 1 \text{ if } a \neq b \\ 0 \text{ if } a = b \end{cases}$

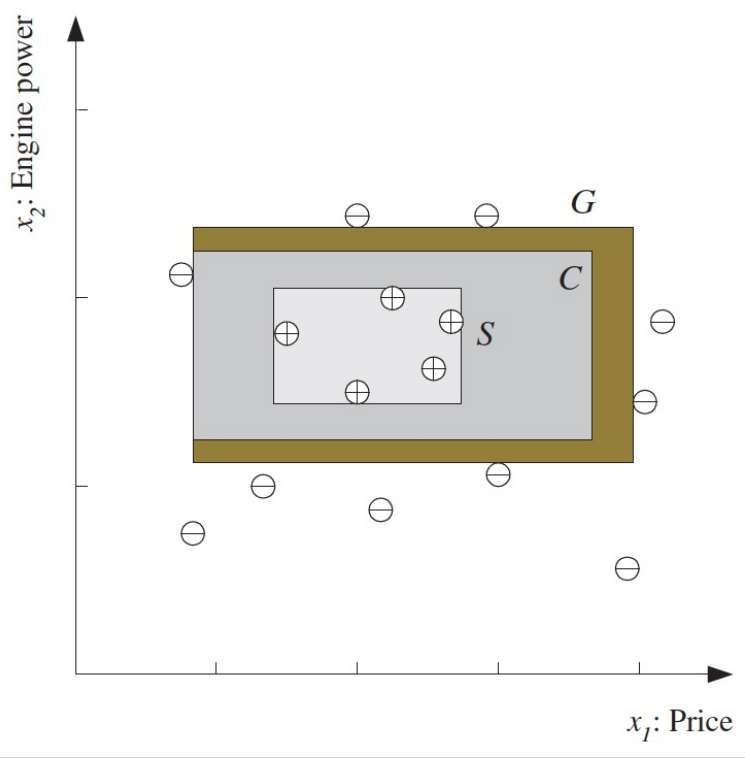
- In our example, $\mathcal{H}$ is the class of all rectangles, each quadruple $\left(p_1^h, p_2^h, e_1^h, e_2^h\right)$ defines an hypothesis h.
  - What is the **"best"** one?

Hypothesis selection 3

- Given the training set, $\left(p_1^h, p_2^h, e_1^h, e_2^h\right)$ should include all the positive examples and none of the negative examples.
- If x_1 and x_2 are real-valued, there are **infinite h-s** for which $E = 0$.
- On the other hand, different candidate hypotheses may make different **predictions** for a future example somewhere close to the boundary between positive and negative examples.
- This is the **problem of generalisation**: how well will our hypothesis classify future examples that are not part of the training set?

Hypothesis selection 4

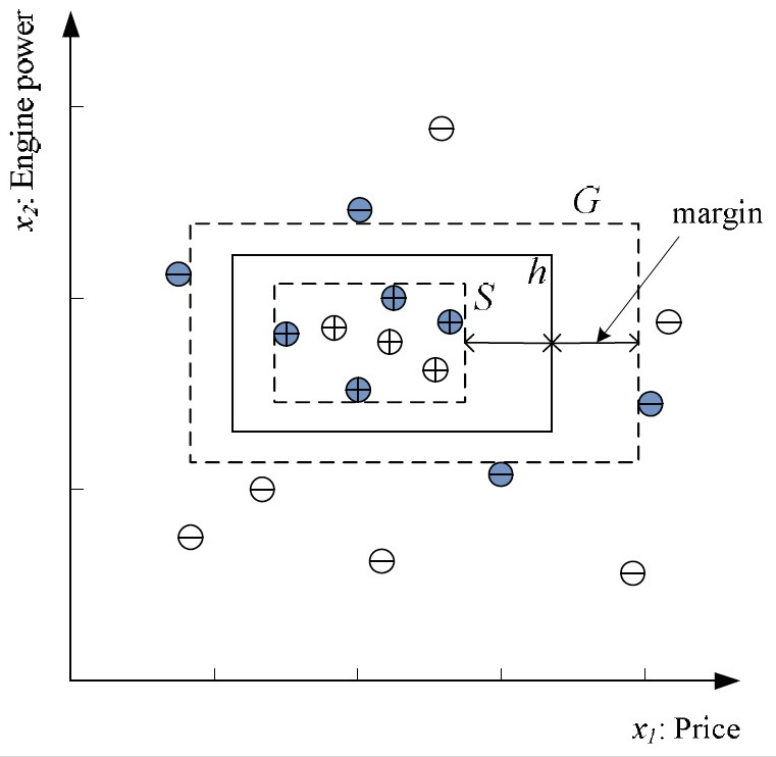
- We call the **most specific hypothesis**, S, the tightest rectangle that includes all the positive examples and none of the negative examples.
  - C may be larger than S but not smaller.
- We call the **most general hypothesis**, G, the largest rectangle that includes all the positive examples and none of the negative examples.
  - C may be smaller than G but not larger.



Hypothesis selection 5

- Any $h \in \mathcal{H}$ between S and G is a valid hypothesis **consistent with the training set**.
  - The set of all possible h consistent with the training set makes up the **version space**.
  - Given another training set, S, G, the version space, and the learned hypothesis h can be different.

- It could be intuitive to choose h **halfway** between S and G.
  - This increases the **margin**, which is the distance between the boundaries of h and the instances closest to it.

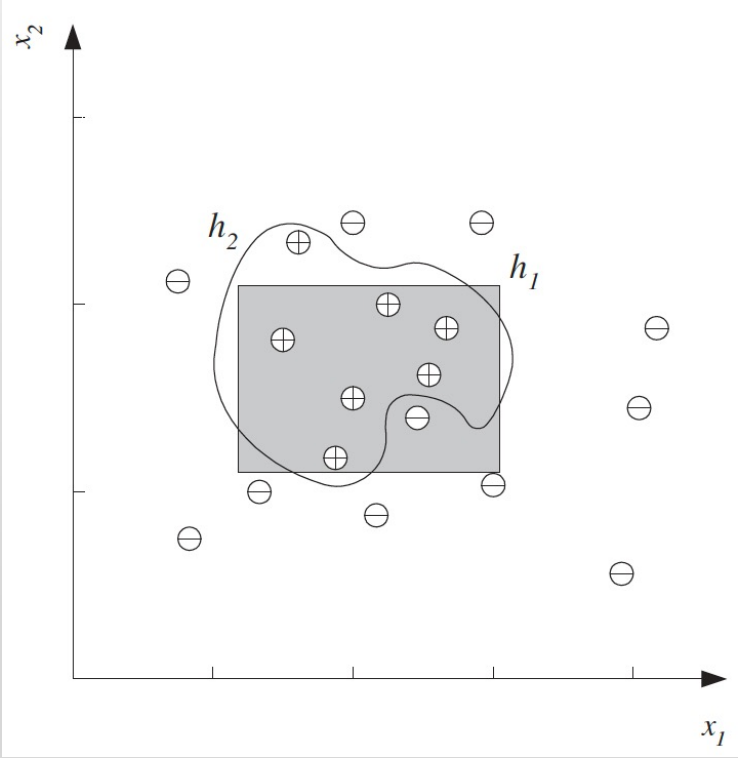


Hypothesis selection 6

- In order to choose h that minimises the empirical error while maximising the margin, we should use an **error** (**loss**) **function** that not only checks whether an instance is on the correct side of the boundary but also how far it is from it.
  - To do that, we need a hypothesis that returns a value carrying a measure of the distance to the boundary and we need a loss function using it, thus different from $E(h|\boldsymbol{\mathcal{X}}) = \sum_{t=1}^{N} 1(h(\boldsymbol{x}_t) \neq r_t)$.
- In some applications, a **wrong decision** may be **very costly**.
  - In such a case, any instance between S and G is a case of **doubt**, which cannot be labelled with certainty due to lack of data.
  - If so, the system **rejects** the instance and defers the decision to a human expert.

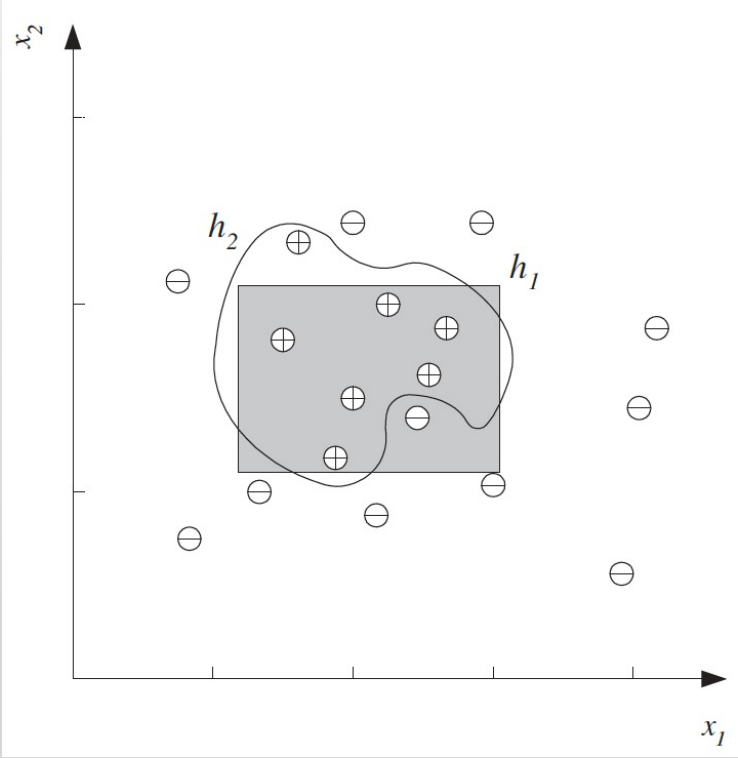
Noise 1

- **Noise** is any unwanted anomaly in the data.
  - Due to noise, a class may be more difficult to learn and **zero error** may be **unattainable** with a simple hypothesis class.
- Possible **causes** of noise:
  - imprecision in recording the input attributes
    - may shift the data in the input space
  - errors in labelling the data
    - may cause labelling of positive instances as negative and vice-versa
    - sometimes called “teacher noise”



Noise 2

- When there is noise, there is **no simple boundary** between the positive and negative instances.
- To separate them, one needs a **more complicated hypothesis**, having a **larger capacity**.
- With a complex model, a fit to the data with zero error can be made.
- Another possibility is to keep the model simple and allow some error.



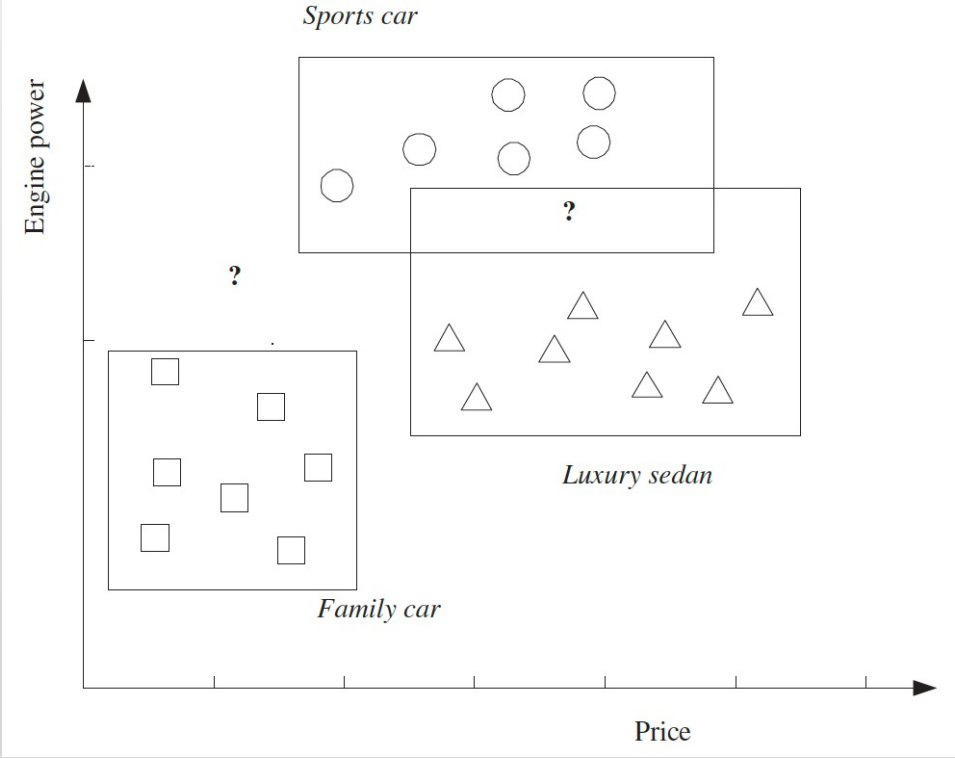
Variance vs. bias

- In our example, using the simple rectangle makes sense because
  - it is easy to check whether a point is inside or outside a rectangle
  - it is **simple to train** as it has few parameters
    - easier to find the corner values of a rectangle than the control points of an arbitrary shape
  - if the training set is small, the rectangle will change less than a complex model with different training instances → a simple model has **less variance**
    - if there is noise, the rectangle is less affected by single instances than a wiggly shape
  - it is **simple to explain**: a rectangle corresponds to defining intervals on the two attributes
- Given comparable empirical error, a simple model will generalise better than a complex model.
  - **Occam's razor**: simpler explanations are more plausible and any unnecessary complexity should be shaved off.
- On the other hand, a simple model is based on **more rigid assumptions** and may fail if the underlying class is not that simple → it has **more bias**.

GRUPPO DI INGEGNERIA CHIMICA DELL'UNIVERSITA'
GRICU
GET INNOVATION

More general classification problems

- In the **general case**, we have K classes $C_i, i = 1, \ldots, K$, and each input instance belongs to one of them.
- In ML for classification, we would like to learn the boundaries separating the instances of one class from those of all the other classes.
- A K-class classification problem can be seen as K two-class problems.
- The training examples belonging to C_i are the positive instances of hypothesis h_i and the examples of all other classes are the negative instances of h_i.



Interpolation and extrapolation

- In classification, given an input, the output is Boolean: yes/no.
- When the output is a numeric value, we would like to learn an (**unknown**) **numeric function** linking it to the input.
- The **training set** of examples drawn from such function is

$$\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_t, r_t\}_{t=1}^{N}$$

  with $r_t \in \mathbb{R} = f(\boldsymbol{x}_t)$.
- If there is **no noise**, the function $f(\boldsymbol{x})$ can be found by **interpolation**.
  - Given N points, a $(N-1)^{\text{th}}$-degree polynomial passes through all the examples in the training set.
- Such function can be used to predict the output for any $\boldsymbol{x}$ out of the training set → **extrapolation**.

Regression 1

- If there is noise acting on the output of the unknown function, we have

$$r_t = f(\boldsymbol{x}_t) + \epsilon_t$$

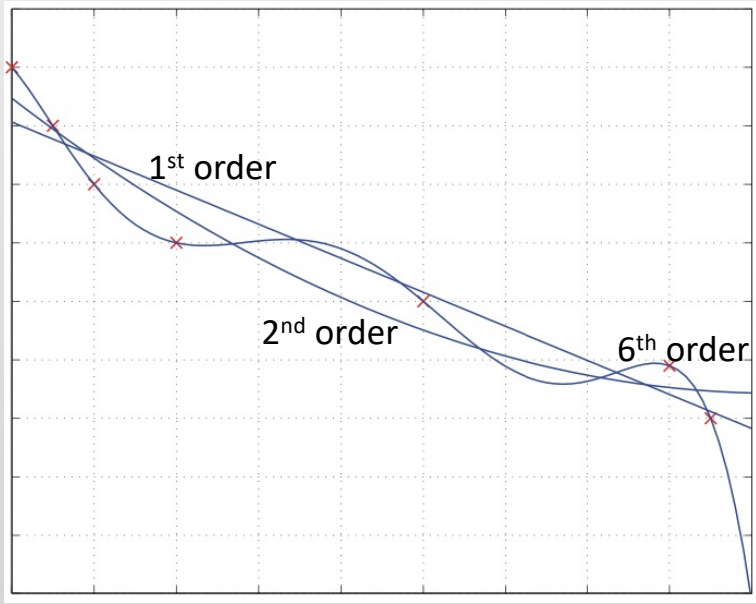
  where ϵ_t represents random noise.
- In this case, we would like to **approximate** the output by a **model** $g(\boldsymbol{x})$ → **regression**.
- A possible reassure of the **empirical error** on the training set $\boldsymbol{\mathcal{X}}$ is

$$E(g|\boldsymbol{\mathcal{X}}) = \frac{1}{N}\sum_{t=1}^{N}[r_t - g(\boldsymbol{x}_t)]^2$$

- Since r_t and $g(\boldsymbol{x}_t)$ are numeric quantities, we can define a **distance** between them.
  - The square of the difference is a possible **loss function**.
  - The absolute value of the difference is another possibility.

Regression 2

- Our aim is to find g that **minimises the loss function**.
- The approach is the same as for classification: we assume a **hypothesis class** for g with a small number of parameters, e.g., a linear function.
- If the linear model is too simple, the empirical error corresponding to the minimization of the loss function will be still too large: in such a case, the output may be taken as a **higher-order function** of the input, e.g., quadratic.
- When the order of the polynomial is increased, the error on the training data decreases.
- On the other hand, a high-order polynomial follows individual examples instead of capturing the general trend.
- **Occam's razor** applies also in the case of regression and we should be careful when fine-tuning the model complexity to match it with the complexity of the function underlying the data.



ML is an ill-posed problem

- Since the training set contains only a small subset of all possible instances, the solution is not unique → ML is an **ill-posed problem**.
  - Such issue appears both in classification and regression applications.

- As training examples increase, we learn more about the underlying function and we can remove inconsistent hypotheses from the hypothesis class.
  - However many consistent hypotheses are still there.

Inductive bias

- Since learning is ill-posed and data by themselves are not sufficient to find the solution, we should make some **extra assumptions** to have a unique solution.
- The set of assumptions we make to perform ML is called the "**inductive bias**" of the learning algorithm.
  - In the classification of cars, there are infinite ways of separating positive from negative examples.
    - Assuming the shape of a rectangle is an inductive bias.
    - Choosing the rectangle with the largest margin is another inductive bias.
  - In regression, there are infinite functions that could relate input to output.
    - Assuming a linear function is an inductive bias
    - Choosing the one that minimizes the squared error is another inductive bias.

Model capacity

- Each hypothesis class has a certain **capacity** and can learn only certain functions.
- The set of functions that can be learned can be extended by using a hypothesis class with larger capacity, containing more complex hypotheses.
  - In classification, a hypothesis class that is the union of two rectangles has higher capacity than a single rectangle, but its hypotheses are more complex → more parameters are needed.
  - In regression, as we increase the order of the polynomial, the capacity and complexity increase.
- The question is to decide where to stop.

Model selection

- Learning is not possible without inductive bias: how to choose the **“right” bias**?
  - This is called **model selection**.
- To answer, we should remember that the aim of ML is the prediction for new cases rather than the replication of the training data.
- We would like to be able to generate the right output for an input instance outside the training set.
- The measure of how well a model trained on a training set predicts the right output for new instances is called **generalisation**.

Model generalisation

- For the best generalisation, we should **match the complexity** of the hypothesis class $\mathcal{H}$ with the complexity of the function underlying the data.
- If $\mathcal{H}$ is less complex than the function, we have **underfitting**.
  - For example, if we fit a straight line to data sampled from a third-order polynomial.
  - In such a case, if we increase the model complexity, the training error decreases.
- If we choose a too complex $\mathcal{H}$, the data is not enough to constrain it and we may end up with a bad hypothesis $h \in \mathcal{H}$. This is called **overfitting**.
  - For example, if we fit two rectangles to data sampled from one rectangle.
  - If there is noise, an overcomplex hypothesis may learn not only the underlying function but also the noise in the data, yielding a bad fit.
- Given a training set and an hypothesis class $\mathcal{H}$, we can find $h \in \mathcal{H}$ that minimises the training error, but, if $\mathcal{H}$ is not chosen well, we will not have good generalization no matter what h we pick.

Triple trade-off

- In all ML algorithms that are trained from example data, there is a **triple trade-off**[4] among
  - the capacity of the hypothesis class (i.e., the complexity of the hypothesis we fit to data),
  - the amount of training data,
  - the generalisation error on new examples.
- As the amount of training data increases, the generalisation error decreases.
- As the complexity of the model class increases, the generalisation error first decreases, then it starts to increase.

Cross-validation 1

- We can measure the **generalisation ability** of a hypothesis if we have access to data outside the training set.
- We simulate this by dividing the dataset we have into two parts:
  - one part is used for training, i.e., to fit a hypothesis,
  - the remaining part is called the **validation set** and is used to test the generalisation ability.
- Given a set of possible hypothesis classes $\mathcal{H}_i, i = 1, \ldots, K$, for each we fit the best $h_i \in \mathcal{H}_i$ on the training set.
- Assuming large enough training and validation sets, the most accurate h_i on the validation set is the best one.
- This process is called **cross-validation**.

Cross-validation 2

- Let us consider, for example, the problem of finding the right order in polynomial regression.
- Given K candidate polynomials of different orders (corresponding to $\mathcal{H}_i, i = 1, \ldots, K$), we fit the coefficients of such polynomials on the training set.
- We calculate their errors on the validation set and take that with less validation error as the best polynomial.
- If we want to give an idea about the expected error of our best model, we should not use the error on the validation set, since this has become a part of the training set → we need a third dataset, said **test set** (or **publication set**), containing examples not used in training nor validation.

Cross-validation 3

- We should always remember that the training data we use are a random sample.
- Given an application, if we collect data again, we will get a slightly different dataset, a slightly different fitted h, and a slightly different validation error.
  - Or, if we have a fixed set which we divide for training, validation, and test, we will have different errors depending on how we do the division.
- We have to check if these differences are significant or due to chance.
- In choosing between two hypothesis classes $\mathcal{H}_i$ and $\mathcal{H}_j$, we will use both of them multiple times on different training and validation sets and check if the difference between average errors of h_i and h_j is larger than the average difference among multiple h_i.

An application to chemical engineering

- The Shell oil company has been collecting values of its operational variables in real time for decades → more than 10^7 data per minute are collected.
- In 2018[5], a model was fit to valve historical process data from two manufacturing sites in order to predict valves' behaviour.
- When the measured data of a valve did not match with the predicted behaviour, the valve was predicted to fail (and it actually happened).
- The model linking process inputs to outputs was not derived on a first-principle basis, yet it worked very well due to the large amount of data.

References

1. Samuel A.L., Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 44, 206–226, 1959
2. Mitchell T., Machine Learning, McGraw-Hill, 1997
3. Alpaydin E., Introduction to Machine Learning 3° Edition, The MIT Press, 2014
4. Dietterich T.G., Machine Learning. In *Nature Encyclopedia of Cognitive Science,* London: Macmillan, 2003
5. Pandya D., Lam B., Suursula S., Kwaspen P., Digital Twins for Predicting Early Onset of Failures Flow Valves. *Proceedings of the AIChE Spring Meeting 2018*

Those who are interested in taking a little **self-assessment quiz** can do that through the Kahoot! App with Game PIN 09442651.