Machine Learning

Session 1 Introduction

- Subject Description and evaluation
- Introduction ML and AI
- Building Blocks: Data, Model and Learning Algorithm
- Paradigms and types of ML problems
- Example: Polynomial Regression (generalization, cross-validation, curse of dimensionality)
- ML in the Real World: Design Cycle
- Organization of the course

Bibliography:

From Bishop , CM Pattern Recognition and Machine Learning. Springer 2006.

Chap 1: 1.1, 1.3, 1.4 ,

https://scikit-learn.org/dev/user_guide.html

Organization

- Theory sessions and lab sessions (seminars and practicals)

G1 English: **Miguel Ángel Cordobés**, Antoine Moulin, Ludovic Schwartz
G2 Català: **Vicenç Gómez**, Emma Fraxanet, Mari Celi Morales, Manuel Portela, MA Cordobés
Groups cannot be changed unless well justified reasons

- Follow **Aula Global** for updates
- Evaluation:

Theory (T): Exam, Midterm and a deliverable. $\mathrm{Exam} \geq 4$

$$\mathbf{T = 0.7 * Exam + 0.2 * Midterm + 0.1 * Deliverable}$$

Midterm: May 12
Deliverable: ~~Jun 14/15~~ **Jun 8**
Midterm and Deliverable can *not* be recovered

Project (P): Programming, in pairs

Only if $\mathrm{P} \geq 5$, the student is eligible for Exam

$$\mathbf{Final\ Grade = 0.7 * T + 0.3 * P}$$

Organization

- Theory sessions and lab sessions (seminars and practicals)

G1 English: **Miguel Ángel Cordobés**, Antoine Moulin, Ludovic Schwartz
G2 Català: **Vicenç Gómez**, Emma Fraxanet, Mari Celi Morales, Manuel Portela, MA Cordobés
Groups cannot be changed unless well justified reasons

- Follow **Aula Global** for updates
- Evaluation:

Resit evaluation:

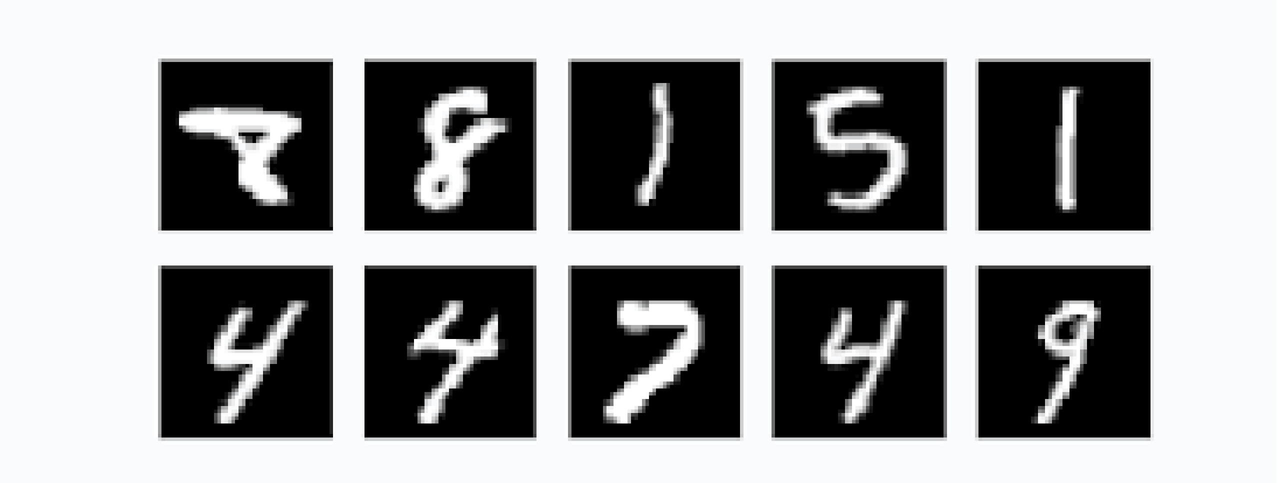
Missing or failed projects **can *not*** be recovered in July
July: 2nd chance $\text{ExamJuly} \geq 5$

$$\textbf{Final Grade July} = \textbf{Max}(\textbf{ExamJuly}, \mathbf{0.7} * \textbf{TJ} + \mathbf{0.3} * \textbf{P})$$

$$\text{TJ} = 0.7 * \text{ExamJuly} + 0.2 * \text{Midterm} + 0.1 * \text{Deliverable}$$

Machine Learning (motivation)

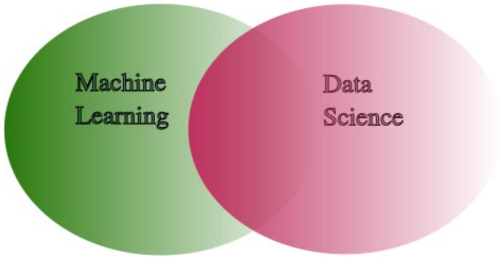
- How would you write a program to classify images like these into corresponding digits?



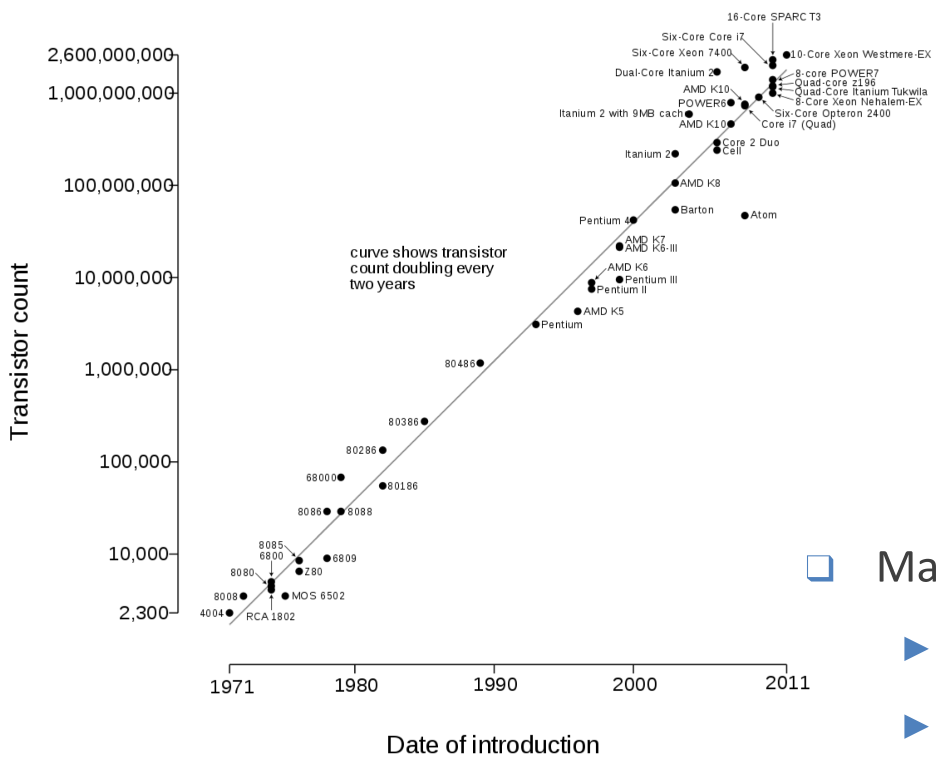
- Machine learning is the **data-driven approach** to generate “intelligent” behavior

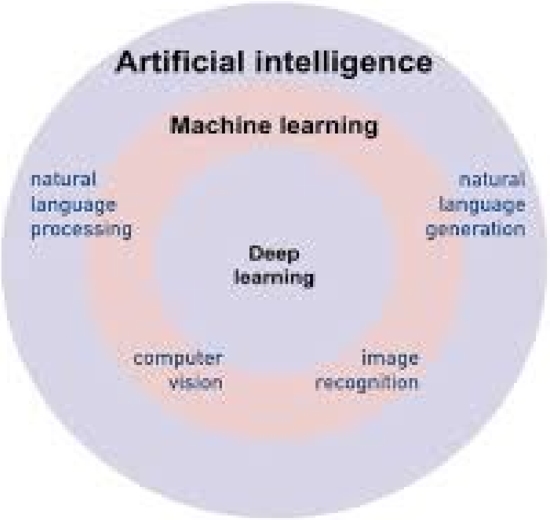
Machine Learning & Data Science & More

- A Moore's law for Data



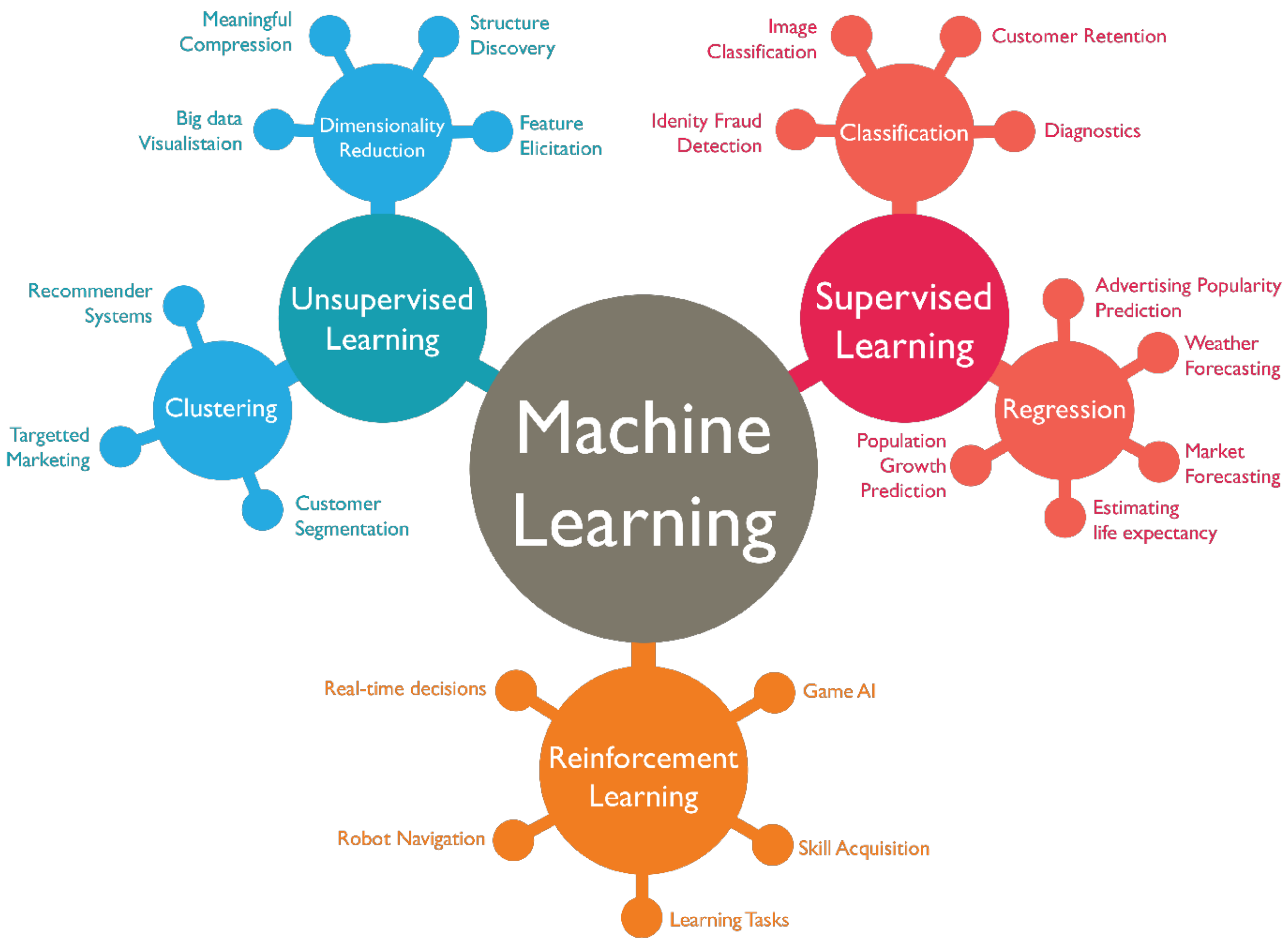
Microprocessor Transistor Counts 1971-2011 & Moore's Law





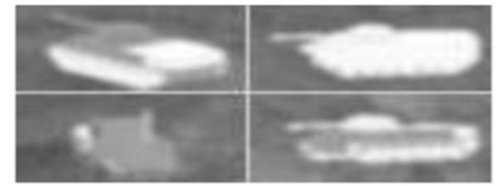
- Main challenges
  - How to build intelligence from these data?
  - Paradigm shift: data is generated with limited control

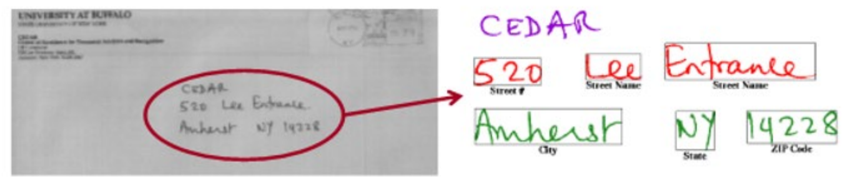
Machine Learning

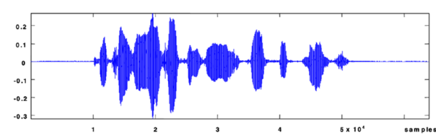


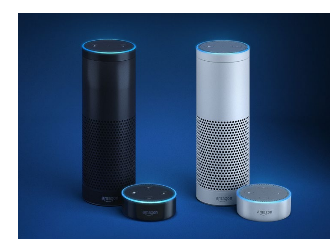
Applications: *almost everywhere*

- **Speech Recognition**:
  - Virtual Personal Assistants:
    - Microphone records acoustic signal
    - Speech Signal is classified into phonemes and/or words
- **Character Recognition**
  - Automated mail sorting
  - Scanner captures image of the text
  - Image is converted into text format
- **Machine Vision**
  - Visual Inspection
  - Imaging device detects ground target
  - Classification cats or dogs
  - Video segmentation
  - Self-driving cars
- **Computer Aided Diagnosis**
  - Medical imaging, EEG, ECG, …
  - Designed to assist (not replace) physicians
- **Product Recommendation:**
  - Store personalized historical data
  - suggest possible products





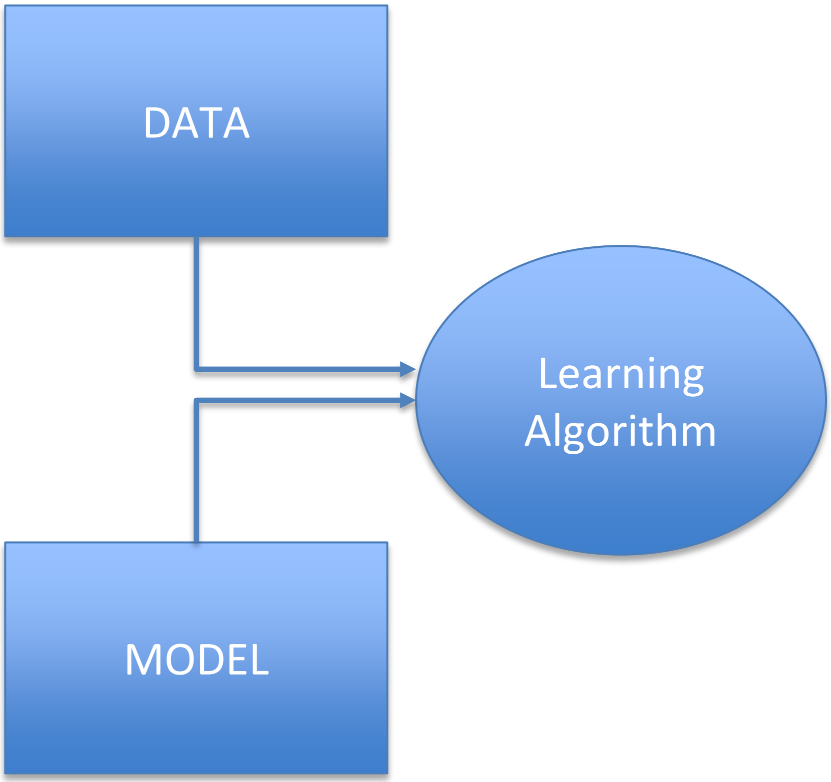




Building blocks

Core elements:

- **Data**
- **Model**
- **Learning Algorithm**



Building blocks: Data

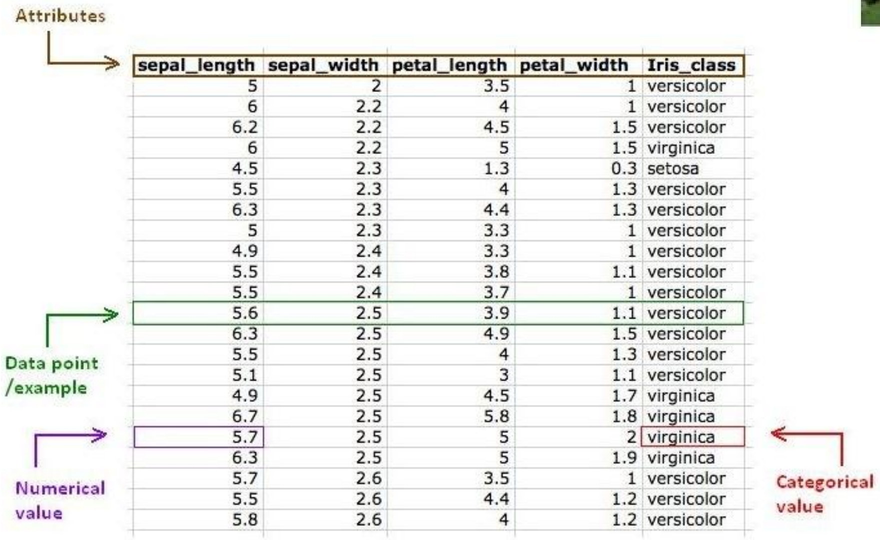
- A dataset is a collection of instances
- Example: the MNIST dataset:

- Each instance represents one out of ten digits
- More than 70,000 instances
- The attributes are binary pixels representing the digit image

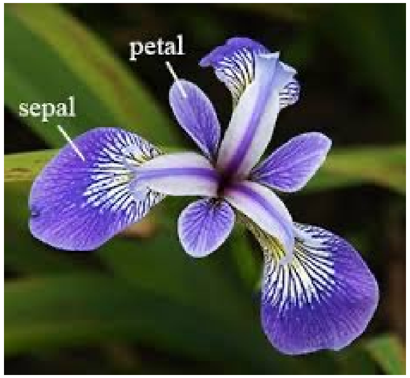
Building blocks: Data

- A dataset is a collection of instances
- Example: the IRIS dataset:

Iris Versicolor Iris Setosa Iris Virginica



sepal_length	sepal_width	petal_length	petal_width	Iris_class
5	2	3.5	1	versicolor
6	2.2	4	1	versicolor
6.2	2.2	4.5	1.5	versicolor
6	2.2	5	1.5	virginica
4.5	2.3	1.3	0.3	setosa
5.5	2.3	4	1.3	versicolor
6.3	2.3	4.4	1.3	versicolor
5	2.3	3.3	1	versicolor
4.9	2.4	3.3	1	versicolor
5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1	versicolor
5.6	2.5	3.9	1.1	versicolor
6.3	2.5	4.9	1.5	versicolor
5.5	2.5	4	1.3	versicolor
5.1	2.5	3	1.1	versicolor
4.9	2.5	4.5	1.7	virginica
6.7	2.5	5.8	1.8	virginica
5.7	2.5	5	2	virginica
6.3	2.5	5	1.9	virginica
5.7	2.6	3.5	1	versicolor
5.5	2.6	4.4	1.2	versicolor
5.8	2.6	4	1.2	versicolor



- Each instance is taken from one out of 3 classes of Iris plant
- Contains 50 instances of each class: 150 examples
- Each example contains 4 (real valued) attributes

Building blocks: Data

– A dataset is a collection of instances

– Example: the Boston Housing dataset:

– Elements n

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	CAT. MEDV
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24	0
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6	0
0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7	1
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4	1
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2	1
0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7	0
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9	0
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1	0
0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5	0
0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9	0
0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15	0
0.11747	12.5	7.87	0	0.524	6.009	82.9	6.2267	5	311	15.2	396.9	13.27	18.9	0
0.09378	12.5	7.87	0	0.524	5.889	39	5.4509	5	311	15.2	390.5	15.71	21.7	0
0.62976	0	8.14	0	0.538	5.949	61.8	4.7075	4	307	21	396.9	8.26	20.4	0
0.63796	0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21	380.02	10.26	18.2	0

– Each instance is a sold house

– 506 examples

– 13 attributes describe the house:

– MEDV is the response (continuous) variable

Table 5.3: Description of Variables for Boston Housing Example

CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 ft^2
INDUS	Proportion of nonretail business acres per town
CHAS	Charles River dummy variable (= 1 if tract bounds river; = 0 otherwise)
NOX	Nitric oxide concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil/teacher ratio by town
B	$1000(\mathrm{Bk} - 0.63)^2$ where Bk is the proportion of blacks by town
LSTAT	% Lower status of the population
MEDV	Median value of owner-occupied homes in $1000s

Building blocks: Data

- The Data Source:
  - Using mathematical notation, we write a dataset $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, t^{(n)}\right)\right\}_{n=1}^{N}$

    where
    - $\boldsymbol{x}^{(n)}$ is the vector of D attributes of the n-th instance $\boldsymbol{x}^{(n)} = [x_1, x_2, \dots, x_D]^\top$
    - $t^{(n)}$ is the response/target/label variable of the n-th instance

  - We often consider a $N \times D$ **design matrix** X and a $N \times 1$ output vector
    - Example ($N = 5$ instances, $D = 3$ attributes):

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} \\ x_1^{(4)} & x_2^{(4)} & x_3^{(4)} \\ x_1^{(5)} & x_2^{(5)} & x_3^{(5)} \end{bmatrix}, \qquad \boldsymbol{t} = \begin{bmatrix} t^{(1)} \\ t^{(2)} \\ t^{(3)} \\ t^{(4)} \\ t^{(5)} \end{bmatrix}$$

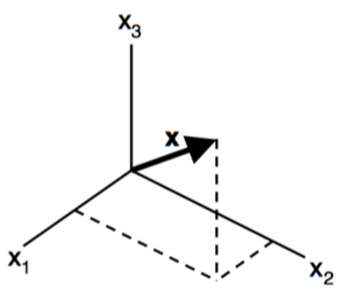
Building blocks: Data

- **Geometrical interpretation of the input space**

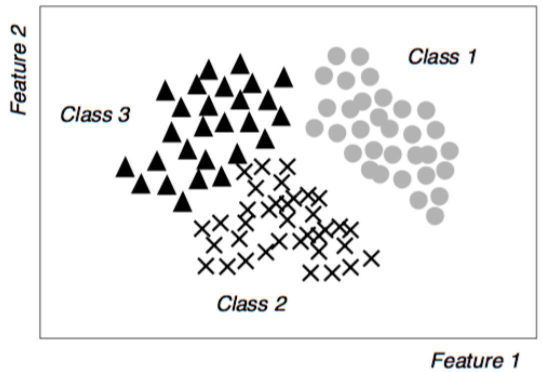
Feature vector : the combination of D features is a column vector (row of design matrix)
Feature space : the D-dimensional space defined by the feature vector
Instances are represented a points in feature space

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_D \end{bmatrix}$$

Feature vector



Feature space (3D)



Scatter plot (2D)

Building blocks:
Data

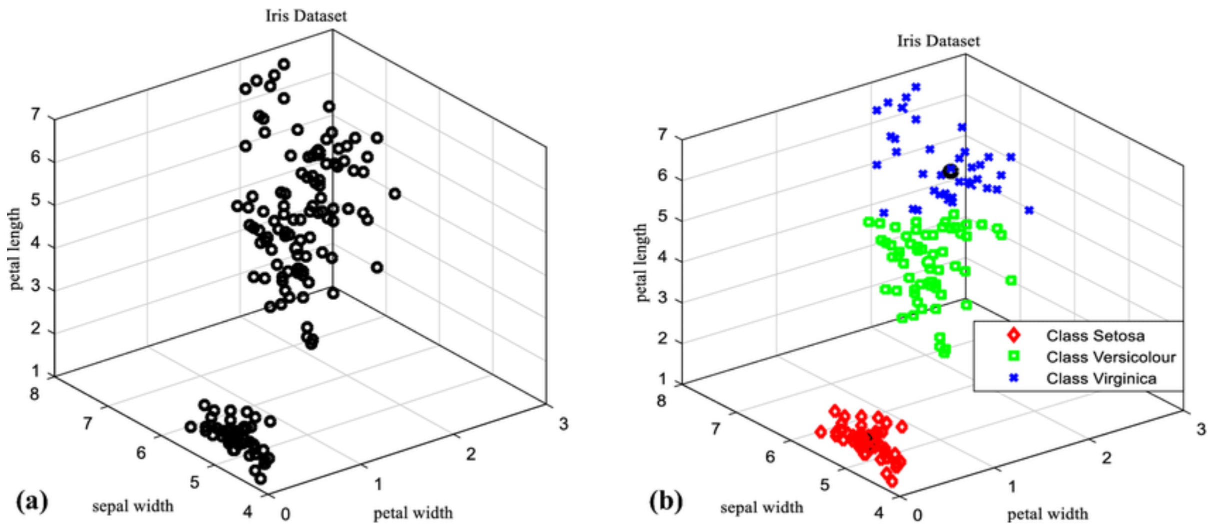
- **Geometrical interpretation of the input space**

Feature vector : the combination of D features is a column vector (row of design matrix)

Feature space : the D-dimensional space defined by the feature vector

Instances are represented a points in feature space

3D feature space of the Iris Dataset (only three out of four features)



Building blocks:
Data

- **Geometrical interpretation of the input space**

Feature vector : the combination of D features is a column vector (row of design matrix)

Feature space : the D-dimensional space defined by the feature vector

Instances are represented a points in feature space

Higher dimensions:

“To deal with a 14-dimensional space, visualize a 3D space and say ‘fourteen’ to yourself very loudly. Everyone does it.”

Geoff Hinton

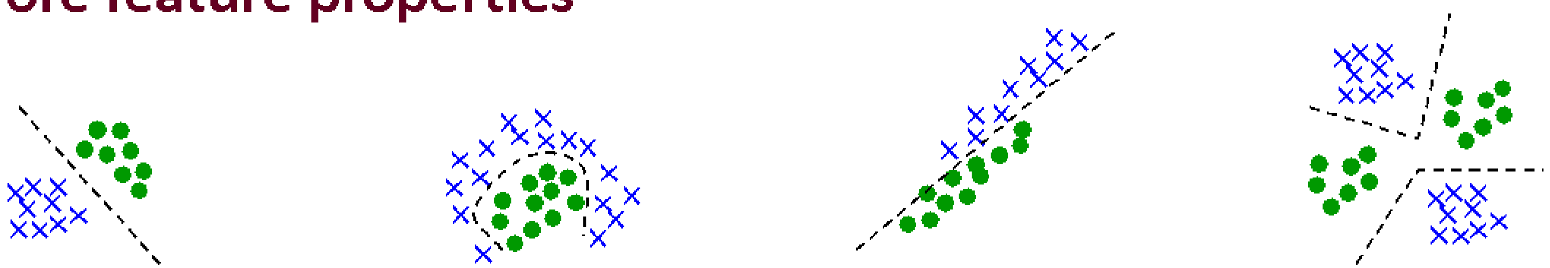
Building blocks: Data

What makes a "good" feature vector?

- The quality of a feature vector is related to its ability to discriminate examples from different classes
  - Examples from the same class should have similar feature values
  - Examples from different classes have different feature values

"Good" features *"Bad" features*

More feature properties



Linear separability *Non-linear separability* *Highly correlated features* *Multi-modal*

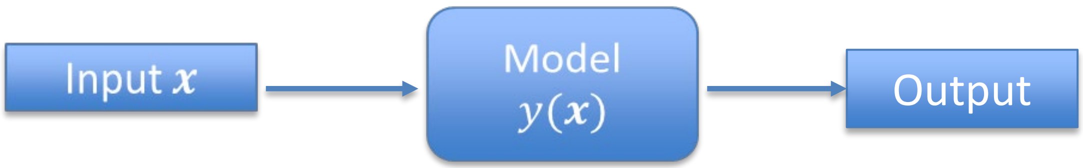
Building blocks: Data

- **The Data Source**
  - We assume that data is generated from an *unknown* process $P(\boldsymbol{x}, t)$
  - Generates instances according to some
    - *unknown* input probability distribution $P(\boldsymbol{x})$
    - *unknown* noisy target function $P(t|\boldsymbol{x})$
  - Important Assumptions:

1. The instances are independent and identically distributed (i.i.d.)
2. The distribution $P(\boldsymbol{x}, t)$ does not change (stationary)

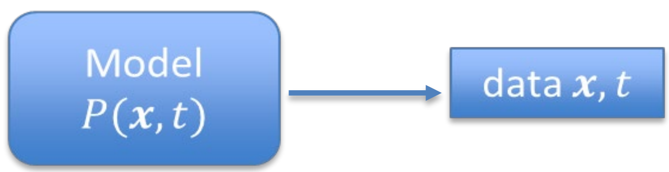
Building blocks: The Model

A function that maps inputs to outputs (**discriminative**)



$y\colon \mathbb{R}^D \rightarrow \mathbb{R}$

A model can also be a processes that generates data (**generative**)



The **Goal** of Machine Learning is to **learn the Model from Data**

Building blocks: The Model

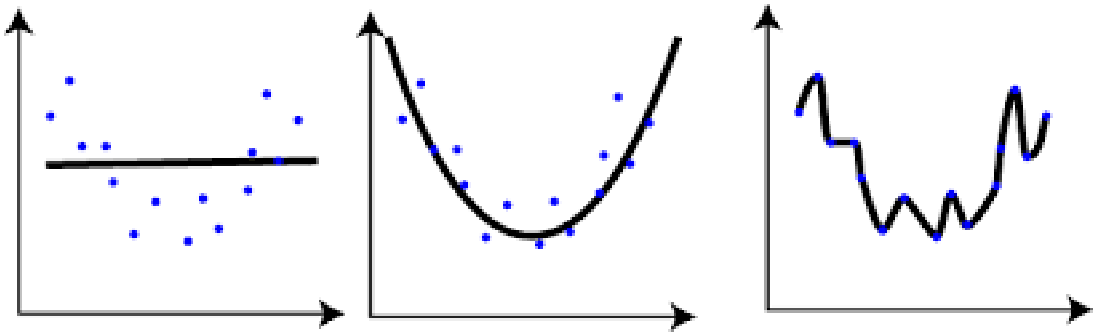
- **The Model**
  - **Makes predictions** t^{N+1} for an unseen data instance $\boldsymbol{x}^{N+1}$
  - We write $y(\boldsymbol{x}, \boldsymbol{w})$ to indicate dependence on model parameters $\boldsymbol{w}$
  - **Example**: a linear model

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{d=1}^{D} w_d \, x_d + w_0 = \boldsymbol{w}^\top \boldsymbol{x} + w_0$$

- **Challenge**
  - What is the correct model we need for our application?

Building blocks: The Model

- Model capacity
  - A measure of the complexity of a model to represent patterns
- Example:



Data in blue, **model** in black
(left) Constant model (low capacity)
(middle) Quadratic model (intermediate capacity)
(right) Arbitrary non-linear model (high capacity)

Building blocks:
The Learning Algorithm

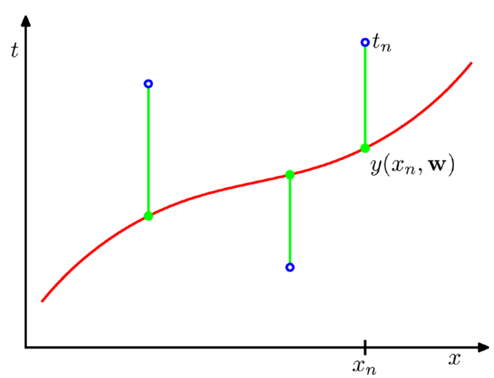
- **The Learning Algorithm**

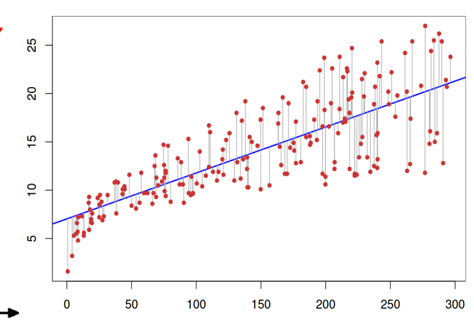
  – A model can **be trained (learn)** from data through parameters $\boldsymbol{w}$

Learning as Optimization → Minimize an error (loss) function

Example: quadratic loss. For a data instance ($\boldsymbol{x}^{(n)}, t^{(n)}$)

$$E_n(\boldsymbol{w}) = \left(t^{(n)} - y\left(\boldsymbol{x}^{(n)}, \boldsymbol{w}\right)\right)^2$$





Building blocks:
The Learning Algorithm

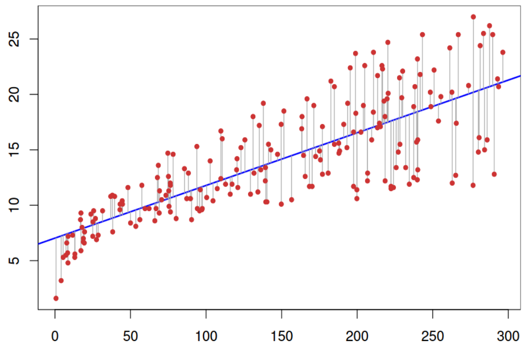
- **The Learning Algorithm**

Learning as Optimization → Minimize an error (loss) function

Example: quadratic loss.

For a dataset $D = \{(\boldsymbol{x}^{(n)}, t^{(n)})\}_{n=1}^{N}$, we sum individual errors (i.i.d. assumption)

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) = \sum_{n=1}^{N} \left(t^{(n)} - y(\boldsymbol{x}^{(n)}, \boldsymbol{w})\right)^2$$



Find $\boldsymbol{w}$ such that $E(\boldsymbol{w})$ is minimized: $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} E(\boldsymbol{w})$

Also known as *training* or *parameter estimation*

Building blocks: The Learning Algorithm

- **The Learning Algorithm**
  - The goal of a learning algorithm is **generalization:**

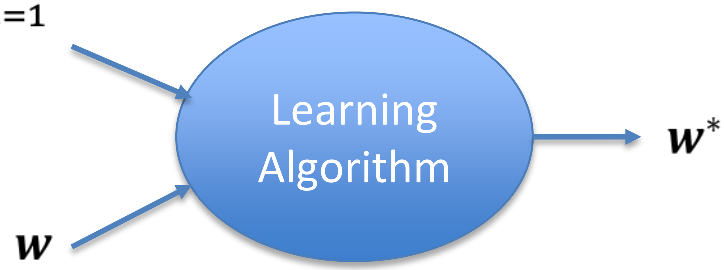
    *From training data D, find parameters $\boldsymbol{w}^*$ such that the model $y(\boldsymbol{x}, \boldsymbol{w}^*)$ performs well on **unseen** data*

- **Challenges**
  - How do we search over the space of parameters $\boldsymbol{w}$?
  - How do we measure good performance in unseen data?

Building blocks:
The Learning Algorithm

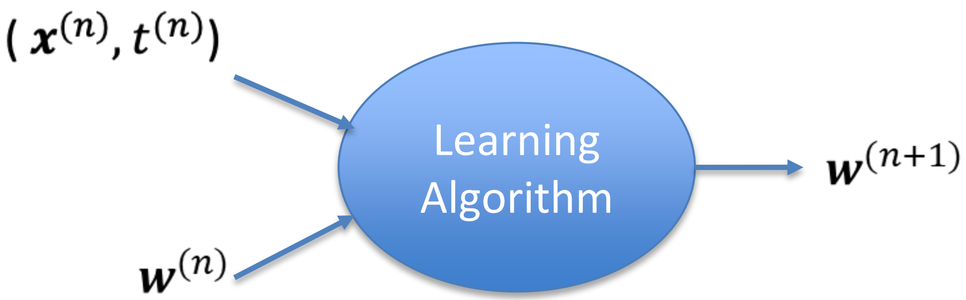
- **The Learning Algorithm**
  - **Batch Learning**
    - Looks at all the N data points in the dataset to train a model
    - Unfeasible if N is large

$D = \{(\boldsymbol{x}^{(n)}, t^{(n)})\}_{n=1}^{N}$



Building blocks: The Learning Algorithm

- **The Learning Algorithm**
  - **Online (sequential) Learning**
    - Processes an instance at a time and then discards
    - Ideal for streaming applications



Approaches to Machine Learning

- **Supervised Learning**

  The data source provides inputs $\boldsymbol{x}^{(n)}$ outputs $t^{(n)}$pairs

  The task is to generalize and predict outputs from unseen input

- **Unsupervised Learning**

  The data source provides one inputs $\boldsymbol{x}^{(n)}$

  The task is to learn about the data generation process

- **Reinforcement Learning**

  The data source (environment) provides feedback (reward) to actions

  The task is to find the actions that maximize long-term reward

Types of ML problems

Machine Learning Problems

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

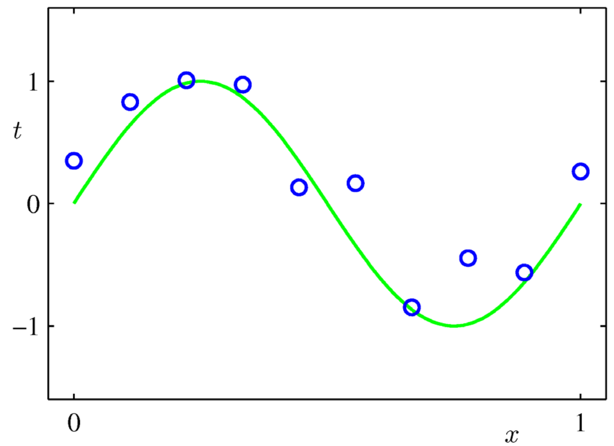
- **Classification**
  - The output is an integer or a probability distribution
    - *Classifying cats vs dogs, genre of a song, SPAM, faces, …*
- **Regression**
  - The output is a real-value number or vector
    - *Predicting the temperature, the share value of a firm, the location of a robot, …*
- **Clustering**
  - The result is a grouping of the data in some meaningful way
    - *Grouping products by some similarity, organizing life forms into a taxonomy of species, …*
- **Structure Learning**
  - The result is a structured representation of the input in terms of primitives
    - *Parsing a sentence, obtaining a symbolic description of an image, …*
- **Dimensionality Reduction**
  - The result is a more compact representation of the input
    - *Getting rid of non-informative input dimensions, eliminating redundant variables, …*

An example : Polynomial regression

- Approximate N datapoints $(x_n, t_n),\ n = 1, \dots, N$ using a **polynomial function**

- In this case, we create our data source as
  - One-dimensional input in the rage $x \in [0,1]$
  - Sinusoidal target plus small Gaussian noise

$$t_n = \sin(2\pi x_n) + \epsilon_n, \qquad \epsilon_n \sim \text{Gaussian}(0, 0.3)$$



An example : Polynomial regression

Note this notation for polynomials

- Approximate N datapoints $(x_n, t_n), n = 1, \dots, N$ using a **polynomial function**

- Our polynomial function for one instance $x^{(n)}$

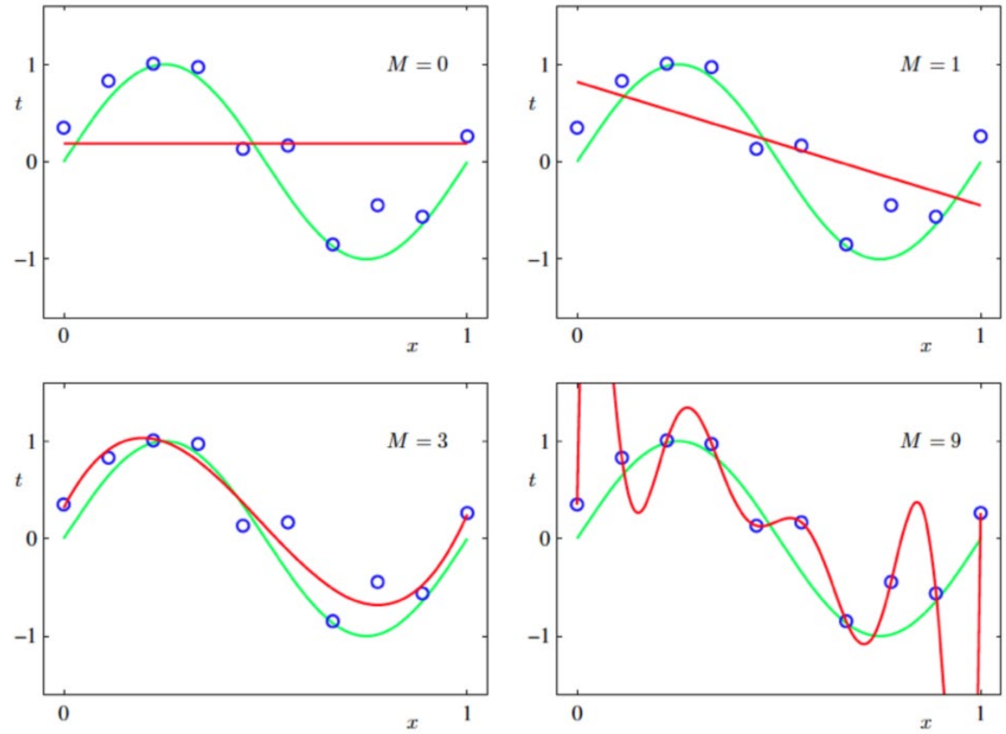
$$y(x_n, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M$$
$$= \sum_{j=0}^{M} w_j x^j$$

  - M is the degree of the polynomial (it is a *hyper-parameter*)
    - Large M means **more** features and a **complex** model (more parameters)
    - Small M means **less** features and a **simple** model (less parameters)

An example : Polynomial regression

Which polynomial should we use?

Images from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)

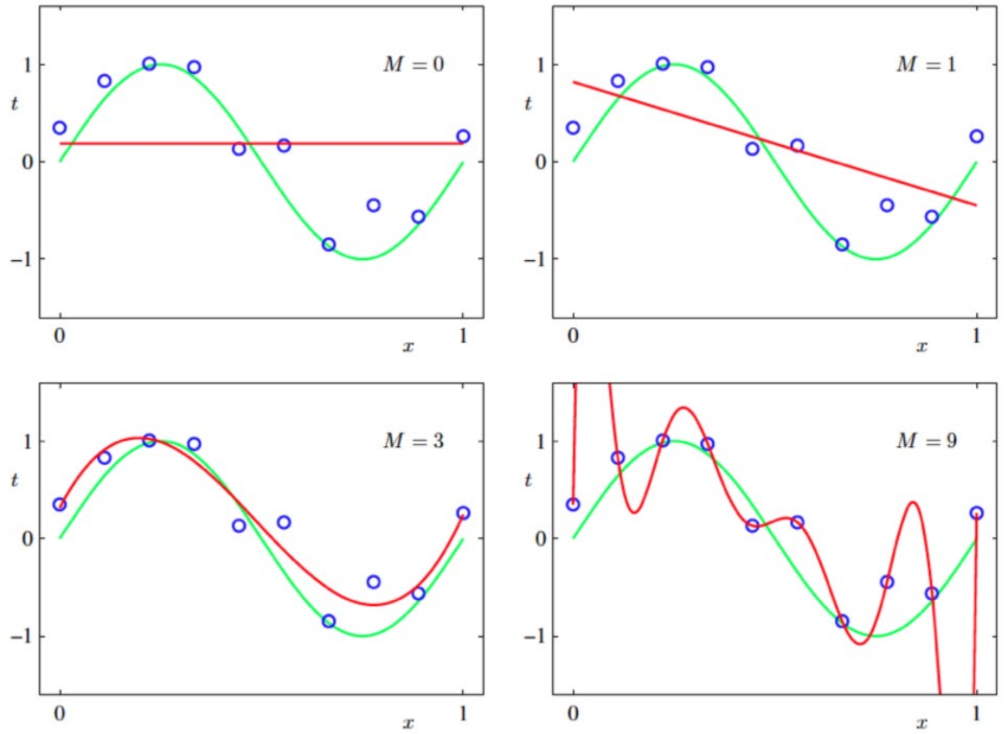


- Think about the error for each value of M
- Is the solution with less error the best one?

An example : Polynomial regression

Which polynomial should we use?

Images from Christopher M Bishop: Pattern Recognition and Machine Learning (PRML)



- **Challenge : How to choose *M*?**
  - Too small M is too restricted to capture the underlying data generator
  - Too large M is too flexible and can easily *overfit* to the seen data

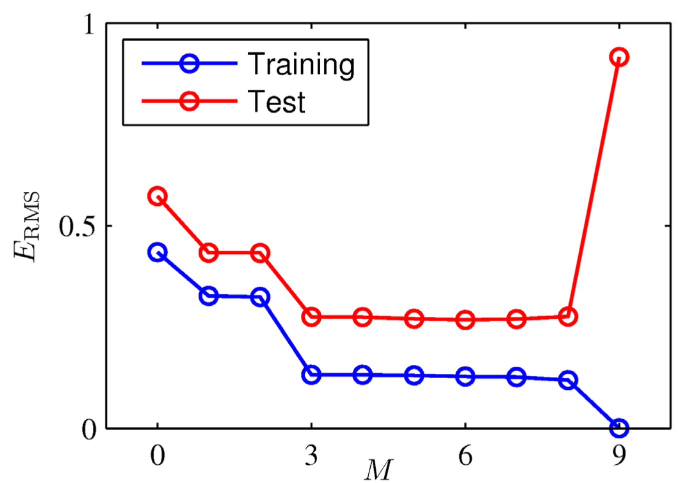
An example : Polynomial regression

- **Challenge : How to choose *M*?**
  - Too small M is too restricted to capture the underlying data generator
  - Too large M is too flexible and can easily *overfit* to the seen data

- Best generalization occurs at an **intermediate** value of M
- Performance on training is **not** a good indicator for generalization
- Consider a separate **test** set not used during ***training***

- Root mean square (RMS) error

$$E_{RMSE}(\boldsymbol{w}) = \sqrt{\sum_{n=1}^{N} \frac{(y(\boldsymbol{x}^{(n)}, \boldsymbol{w}) - t^n)^2}{N}}$$



What is our best model in this case? -> M=3

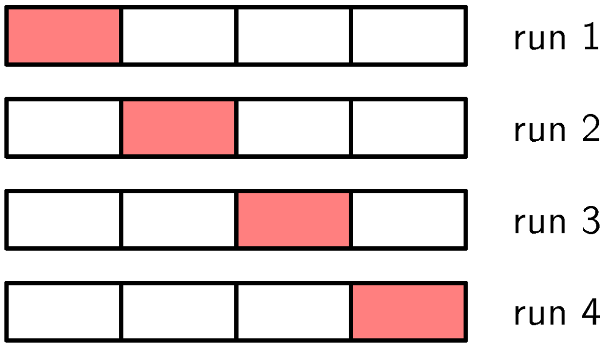
Why? The most simple with the minimum error

An example : Polynomial regression

- **Challenge : How to choose *M*?**
  - Too small M is too restricted to capture the underlying data generator
  - Too large M is too flexible and can easily *overfit* to the seen data

- **Cross-Validation**: A technique to assess generalization
  1. Split full dataset in S folds
  2. Train with $S - 1$ and test with S
  3. Repeat S times
  4. Take the mean of the S performances

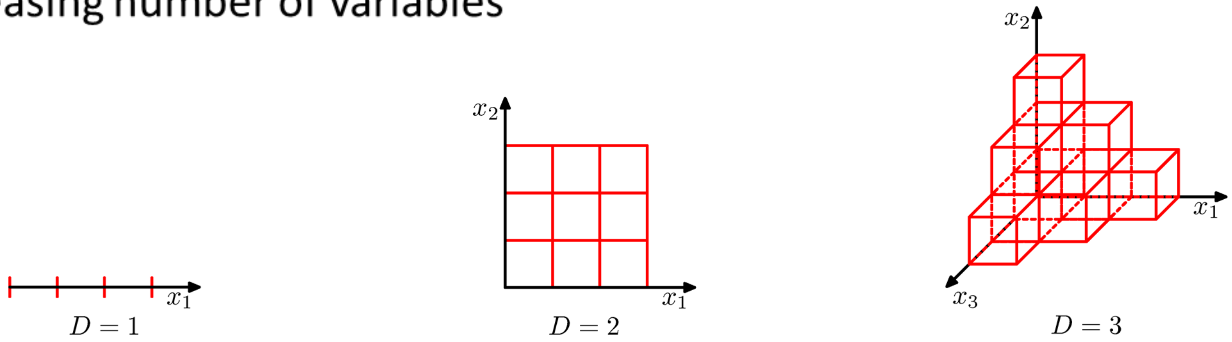
- Example: $S = 4$



An example : Polynomial regression

- **Challenge:** How does our problem scale with D?

- **The curse of dimensionality**: the difficulty increases **exponentially** with an increasing number of variables

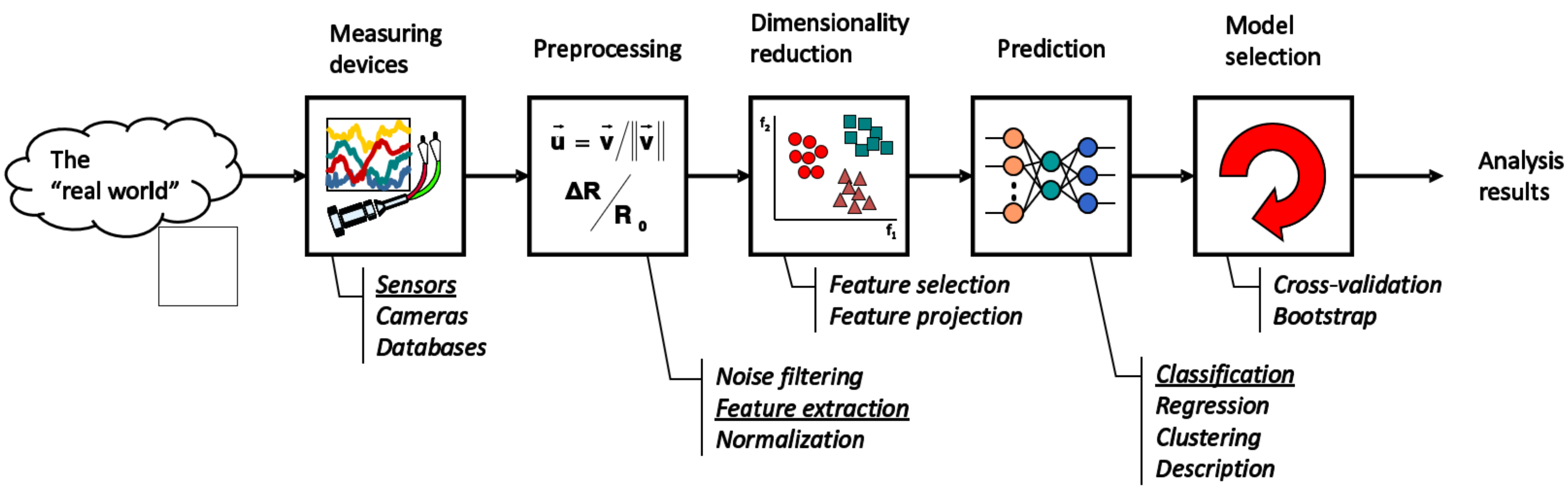


- In practice, real data
  - is often confined to a region having **lower effective dimensionality**
  - **Local Smoothness** : does not change much locally

- Machine Learning systems exploit these two properties

Machine Learning system in the real-world

- **A basic machine learning system contains**
  - A sensing device (receives input data)
  - A preprocessing mechanism
  - A feature extraction mechanism (manual or automated)
  - A statistical model for outputs (classification or regression)
  - A set of examples (training set) already labeled



A Machine Learning design cycle

- **Data Collection**
  - Ideally select data which is informative for the task
  - Number of training examples, balanced vs non-balanced dataset, number of classes, etc., …
  - Challenges: sometimes task is not even known at the data collection time
- **Preprocessing**
  - Feature selection, dimensionality reduction, …
  - Critical to the success (good data vs big data)
  - Prior knowledge can be used if available
- **Main Approach**
  - What model to use?
  - What are the parameters, hyper-parameters?
- **Training**
  - Adapt the parameters to better explain the data and to predict unobserved data
  - What is the error function or likelihood to optimize?
- **Evaluation**
  - How well does the trained model perform in unseen data?
  - Overfitting vs Generalization
  - How to measure performance? ROC curves, f1score, … (relate with the error function above)

- Possibly refine: collect new data, modify model, etc, …

Organization

- Contents:

T1 : Introduction

T2 : Unsupervised Methods: clustering

T3 : Generative Models: the Gaussian

T4 : Mixture of Gaussians

T5 : Principal Component Analysis

T6 : Ensemble Methods for Classification

T7 : Support vector machines and kernel methods

T8 : Linear models for regression. Regularization

T9 : Linear models for classification

T10 : Deep learning 1

T11 : Deep learning 2

T12 : Deep learning 3