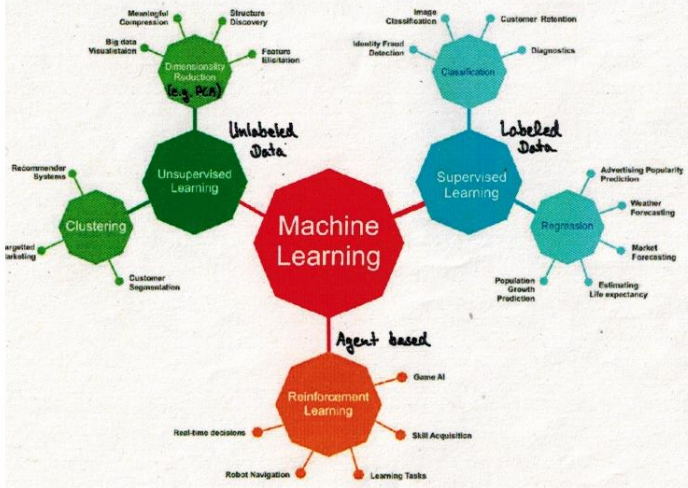
Machine Learning Basics



Assumption: Observed data $\mathcal{D}$ drawn i.i.d. from unknown data-generating process $\mathbb{P}_{xy}$ — independent rows in dataset

(Not always realistic, scenarios like time series are clearly not iid)

Data in Supervised Learning

ML (typically): prediction more important than explaining

Goal: Discover predictive rule behind some (assumed) relationship between target and features — functional & intrinsic pattern

Row 1 and y_1: feature-Vector 1 $(\underline{x}^{(1)}, y^{(1)})$

	Features x				Target y
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
labeled data [training]	4.3	3.0	1.1	0.1	setosa
	5.0	3.3	1.4	0.2	setosa
	7.7	3.8	6.7	2.2	virginica
	5.5	2.5	4.0	1.3	versicolor
unlabeled data [predict]	5.9	3.0	5.1	1.8	?
	4.4	3.2	1.3	0.2	?

labels

n-Zeilen

$\underline{x}_2$ Column 2 / Feature 2

p-Merkmale

Target variable types:

- Numerical ($\mathbb{R}$)
- Integer ($\mathbb{Z}$)
- Categorical ($\{C_1, \dots, C_g\}$)
- Binary ($\{0,1\}$) (int. $\{1,-1\}$)

Numerical, Integer → Regression task; Categorical, Binary → Classification task (then Learner called "classifier"); both → Supervised task

Models & Parameters

A function $f: \mathcal{X} \to \mathbb{R}^g$ is called a **Model** (or **Hypothesis**)

In most Regression tasks $g = 1$. In Classification tasks $g = \#$ Categories and f is e.g. a score or class probability

! ML requires contraining f to a certain type of function which we call **structual prior** (e.g. design choice is linear functions)

NOT after closely looking at data!

The set $\mathcal{H} := \{f \mid f$ belongs to the class of the structual prior$\}$ is called **Hypothesis space / Model class**. $\mathcal{H}$ choice upfront, usually automatically from Learnerchoice

Usually $\mathcal{H}$ is constructed as a parametrized family of curves with parameters $\underline{\theta} := (\theta_1, \dots, \theta_d) \in \Theta$ **Parameter space** $\mathbb{R}^d$

$\Rightarrow \mathcal{H} = \{f_\theta \mid f_\theta$ belongs to a family of curves parametrized by $\theta\}$

Inducer, Learner (-Algorithm) $\mathcal{I}: \mathbb{D} \times \Lambda \to \mathcal{H}$ bzw. Θ, $(\mathcal{D}, \lambda) \mapsto \hat{f}_{\mathcal{D},\lambda}$ (training dataset, hyperparameter control setting)

Therefore: **!** Finding the optimal model is equivalent to finding the optimal parameters ⟹ picks best element of hypothesis space / parameter for given training data

↓ evaluates $f \in \mathcal{H} \mid \Theta$ on $\mathcal{D}$ with risk function $R_{emp}(f)$ or $R_{emp}(\theta)$

uses optimization method to find $\hat{f}$ or $\hat{\theta}$, the model with the lowest risk

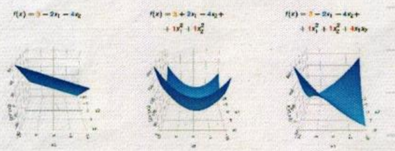
! The parameter-to-model mapping can be non-injective, i.e. one model is descriped by different parameter vectors.

Learning components: $\mathcal{H}$, risk, optimization

Example for Hypothesis/Parameter Spaces

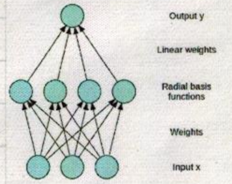
Bivariate quadratic function:

$$\mathcal{H} = \{f : f(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2, \theta \in \mathbb{R}^6\},$$



RBF Network:

$$\mathcal{H} = \left\{ f : f(\mathbf{x}) = \sum_{i=1}^{k} a_i \rho(\|\mathbf{x} - \mathbf{c}_i\|) \right\},$$



$a_i :=$ weight of i-th neuron

$c_i :=$ its center vector

$\rho(\|\vec{x} - c_i\|) := \exp(-\beta \cdot \|\vec{x} - c_i\|^2)$ is the i-th **r**adial **b**asis **f**unction with bandwidth $\beta \in \mathbb{R}$.

Notation

nput/Feature Space: $\mathcal{X}$ (usual $\mathcal{X} \subset \mathbb{R}^p$)

utput/Target Space: $\mathcal{Y}$

All data sets of size n: $\mathbb{D}_n := (\mathcal{X} \times \mathcal{Y})^n$

All finite data sets: $\mathbb{D} := \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$

or the actually observed data we denote:

-th observation: $(\underline{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$ — i-th dataset row

-th feature vector: $\underline{x}_j := (x_j^{(1)}, \dots, x_j^{(n)})^T$ — j-th dataset column

bserved data set: $\mathcal{D} := ((\underline{x}^{(1)}, y^{(1)}), \dots, (\underline{x}^{(n)}, y^{(n)})) \in \mathbb{D}_n$

ata generating process: $\mathbb{P}_{xy} : \mathcal{X} \times \mathcal{Y} \to [0,1]$

ncoding of Categorical Features

Why? Most learning algos can only handle numerical features

et x be a **nominal-scaled** feature, i.e. $x \in \{C_1, \dots, C_k\}$.

e transform the scalar x into a vector: $\sigma(x) := \begin{bmatrix} \mathbb{1}_{\{C_1\}}(x) \\ \vdots \\ \mathbb{1}_{\{C_{\tilde{k}}\}}(x) \end{bmatrix} \in \{0,1\}^{\tilde{k}}$.

ach entry of $\sigma(x)$ is treated as a seperate (binary) feature.

$\Rightarrow \forall x, \sigma(x) = (0 \dots 1 \dots 0)$ — length k

One-hot-encoding: $\tilde{k} = k$ dummies. So exactly one entry of $\sigma(x)$ is 1 ("hot").

Dummy-encoding: $\tilde{k} = k-1$ dummies. So at most one entry of $\sigma(x)$ is 1.

↳ cuts redundancy of one-hot-encoding.

Necessary if non-singular-matrix is required (e.g. Lin.Reg.)

or an **ordinal-scaled** feature x we use an encoding that reflects the ordinality, e.g. a sequence of integer values.