

Machine Learning

Fabrice Rossi

SAMM
Université Paris 1 Panthéon Sorbonne

2018

Standard programming

Solving a task with a computer

- ▶ input and output definition
- ▶ algorithm design
- ▶ implementation

Examples

- ▶ jpeg image converter
- ▶ interactive 3D world
- ▶ computational fluid dynamics
- ▶ chess program
- ▶ spam filtering (with e.g. [SpamAssassin](#))

Standard programming

Solving a task with a computer

- ▶ input and output definition
- ▶ algorithm design
- ▶ implementation

Examples

- ▶ jpeg image converter
- ▶ interactive 3D world
- ▶ computational fluid dynamics
- ▶ chess program
- ▶ spam filtering (with e.g. SpamAssassin)

100% human design

Machine programming

- ▶ climbing one step in abstraction
- ▶ designing programs **with a program**
- ▶ output: a program that solves a task
- ▶ input?

Machine programming

- ▶ climbing one step in abstraction
- ▶ designing programs **with a program**
- ▶ output: a program that solves a task
- ▶ input?

Learning from examples

- ▶ input: a set of pairs (input, output)
- ▶ output: a program g
- ▶ if (\mathbf{x}, \mathbf{y}) is in the set, g should output \mathbf{y} if given \mathbf{x} as input (i.e., $g(\mathbf{x}) = \mathbf{y}$)
- ▶ this is called **supervised learning**
- ▶ **example**: produce **SpamAssassin** using tagged emails!

Minimal formal model

Data spaces

- ▶ \mathcal{X} : input space ($\mathbf{X} \in \mathcal{X}$)
- ▶ \mathcal{Y} : output space ($\mathbf{Y} \in \mathcal{Y}$)
 - ▶ if $|\mathcal{Y}| < \infty$: **classification**
 - ▶ if $\mathcal{Y} = \mathbb{R}$: **regression**

Programs

- ▶ mathematical version: a function g from \mathcal{X} to \mathcal{Y}
- ▶ running the program: $g(\mathbf{X}) = \mathbf{Y}$
- ▶ preferred term: **model**

Machine learning program

- ▶ input: a data set $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{1 \leq i \leq N}$ (or a data sequence)
- ▶ output: a function from \mathcal{X} to \mathcal{Y} (a model)

Is machine learning feasible?

Is machine learning feasible?

Yes!

Is machine learning feasible?

Yes!

- ▶ strong theoretical guarantees
- ▶ **Probably Approximately Correct (PAC) framework**
 - ▶ we look for g (the task solving program/model) in some class of models
 - ▶ if the class is not too complex, g can be recovered approximately with high probability given a reasonable amount (N) of input data
- ▶ **Asymptotic framework**
 - ▶ in addition the class of models can grow in complexity with the data size
 - ▶ then the best possible g (no restriction) can be reached **asymptotically** (almost surely in some cases): this is a **consistency** property

Is machine learning feasible?

Yes!

- ▶ strong theoretical guarantees
- ▶ **Probably Approximately Correct (PAC) framework**
 - ▶ we look for g (the task solving program/model) in some class of models
 - ▶ if the class is not too complex, g can be recovered approximately with high probability given a reasonable amount (N) of input data
- ▶ **Asymptotic framework**
 - ▶ in addition the class of models can grow in complexity with the data size
 - ▶ then the best possible g (no restriction) can be reached **asymptotically** (almost surely in some cases): this is a **consistency** property

How?

Additional hypothesis

- ▶ \mathcal{X} is equipped with a dissimilarity d
- ▶ d is a **dissimilarity** on \mathcal{X} iff:
 1. d is a function from $\mathcal{X} \times \mathcal{X}$ to \mathbb{R}^+
 2. $\forall \mathbf{X}, \mathbf{X}', d(\mathbf{X}, \mathbf{X}') = d(\mathbf{X}', \mathbf{X})$
 3. $\forall \mathbf{X}, \mathbf{X}', \mathbf{X} \neq \mathbf{X}' \Leftrightarrow d(\mathbf{X}, \mathbf{X}') > 0$

Neighbors

- ▶ $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$
- ▶ nn is the function from $\mathcal{X} \times \{1, \dots, N\}$ to $\{1, \dots, N\}$ such that
 - (1) $d(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x},1)}) \leq d(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x},2)}) \leq \dots \leq d(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x},N)})$
 - (2) if $d(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x},k)}) = d(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x},k+1)})$ then $nn(\mathbf{x}, k) < nn(\mathbf{x}, k+1)$
- ▶ denoted $nn_{\mathcal{D}}$ if needed

K nearest neighbors

Finite output space

- ▶ when $|\mathcal{Y}| < \infty$
- ▶ given as input the data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$ and the parameter K , the K nearest neighbors (K -nn) machine learning program outputs g_{K-nn} defined by

$$g_{K-nn}(\mathbf{x}) = \arg \max_{\mathbf{y} \in \mathcal{Y}} |\{k \in \{1, \dots, K\} \mid \mathbf{Y}_{nn_{\mathcal{D}}}(\mathbf{x}, k) = \mathbf{y}\}|$$

- ▶ in simple terms: $g_{K-nn}(\mathbf{x})$ is the most common value of \mathbf{y} in the examples that are the K closest ones to \mathbf{x}

K nearest neighbors

Finite output space

- ▶ when $|\mathcal{Y}| < \infty$
- ▶ given as input the data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$ and the parameter K , the K nearest neighbors (K -nn) machine learning program outputs g_{K-nn} defined by

$$g_{K-nn}(\mathbf{x}) = \arg \max_{\mathbf{y} \in \mathcal{Y}} |\{k \in \{1, \dots, K\} \mid \mathbf{Y}_{nn_{\mathcal{D}}}(\mathbf{x}, k) = \mathbf{y}\}|$$

- ▶ in simple terms: $g_{K-nn}(\mathbf{x})$ is the most common value of \mathbf{y} in the examples that are the K closest ones to \mathbf{x}

Consistency

- ▶ if \mathcal{X} is a finite dimensional Banach space and $|\mathcal{Y}| = 2$
- ▶ then the K -nn method is **consistent** if K depends on N , K_N , in such a way that $K_N \rightarrow \infty$ and $\frac{K_N}{N} \rightarrow 0$

Algorithm choice

- ▶ numerous ML algorithms
- ▶ with parameters (e.g., K for the K -nn method)
- ▶ How to choose the “best” model?

Efficiency

- ▶ computational efficiency
- ▶ data efficiency

And many other issues...

Artificial Intelligence

Artificial Intelligence (AI) is intelligence displayed by machines, in contrast with the natural intelligence (NI) displayed by humans and other animals.

Wikipedia AI page

Machine learning

- ▶ is about **learning**:
 - ▶ this is only a small part of intelligence!
 - ▶ a data set is needed: it is produced by humans (limited autonomy)
- ▶ ML is only a tool that might be useful (in the future!) to build real AI
- ▶ **beware of syllogisms**: what *can* be solved with human intelligence does not always *need* intelligence to be solved

Introduction

Loss and risk

Judging a ML algorithm

Supervised learning

- ▶ input: $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{1 \leq i \leq N}$
- ▶ output: $g : \mathcal{X} \rightarrow \mathcal{Y}$
- ▶ “ideally” we would like that $\forall i, g(\mathbf{X}_i) = \mathbf{Y}_i$

Weakening the goal

- ▶ $\forall i, g(\mathbf{X}_i) = \mathbf{Y}_i$ is too strong
 - ▶ limited knowledge
 - ▶ intrinsic randomness
- ▶ approximate answers, i.e. $\forall i, g(\mathbf{X}_i) \simeq \mathbf{Y}_i$

Loss function

A loss function l is

- ▶ a function from $\mathcal{Y} \times \mathcal{Y}$ to \mathbb{R}^+
- ▶ such that $\forall \mathbf{Y} \in \mathcal{Y}, \quad l(\mathbf{Y}, \mathbf{Y}) = 0$

Interpretation

$l(g(\mathbf{X}), \mathbf{Y})$ measures the loss incurred by the user of a model g when the true value \mathbf{Y} is replaced by the value $g(\mathbf{X})$.

Weakening the goal

- ▶ $\forall i, g(\mathbf{X}_i) = \mathbf{Y}_i$ is replaced by
- ▶ $\forall i, l(g(\mathbf{X}_i), \mathbf{Y}_i)$ should be as small as possible

Examples

$\mathcal{Y} = \mathbb{R}$ (Regression)

- ▶ $l_2(p, t) = (p - t)^2$
- ▶ $l_1(p, t) = |p - t|$
- ▶ $l_{APE}(p, t) = \frac{|p-t|}{|t|}$

$|\mathcal{Y}| < \infty$ (Classification)

- ▶ $l_b(p, t) = \mathbf{1}_{p \neq t}$
- ▶ general case when $\mathcal{Y} = \{y_1, y_2\}$

	$t = y_1$	$t = y_2$
$p = y_1$	0	$l(y_1, y_2)$
$p = y_2$	$l(y_2, y_1)$	0

asymmetric costs are important in practice (think SPAM versus non SPAM)

Quality of a model

From local to global

- ▶ loss functions work at a local scale: what happens for *one* input
- ▶ we need a *global* assessment of a model g : how will the model behave if deployed?

Quality of a model

From local to global

- ▶ loss functions work at a local scale: what happens for *one* input
- ▶ we need a *global* assessment of a model g : how will the model behave if deployed?
- ▶ **expected loss**

Quality of a model

From local to global

- ▶ loss functions work at a local scale: what happens for *one* input
- ▶ we need a *global* assessment of a model g : how will the model behave if deployed?
- ▶ **expected loss**

Empirical risk

- ▶ simple aggregation of local losses: average loss
- ▶ the **empirical risk** of a model g on a data set $\mathcal{D} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{1 \leq i \leq N}$ for a loss function l is

$$\hat{R}_l(g, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^N l(g(\mathbf{X}_i), \mathbf{Y}_i) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}} l(g(\mathbf{x}), \mathbf{y})$$

- ▶ a good model has a low empirical risk

Loss functions

- ▶ should not be chosen lightly
- ▶ have complex consequences, for instance
 - ▶ asymmetric losses can be seen as example weighting
 - ▶ APE loss induces underestimation
- ▶ a good data scientist knows how to translate objectives into loss functions

Empirical risk

- ▶ is only an average: does not rule out extreme behavior
- ▶ in particular, the actual loss can strongly vary with the “location” of \mathbf{x} in \mathcal{X}
- ▶ reporting only the empirical risk is not sufficient!

Confusion matrix

- ▶ when \mathcal{Y} is finite, one should report a confusion matrix $\widehat{C}(\mathcal{D})$ with entries

$$\widehat{C}_{\mathbf{y}, \mathbf{y}'} = |\{i \in \{1, \dots, N\} \mid g(\mathbf{X}_i) = \mathbf{y} \text{ and } \mathbf{Y}_i = \mathbf{y}'\}|$$

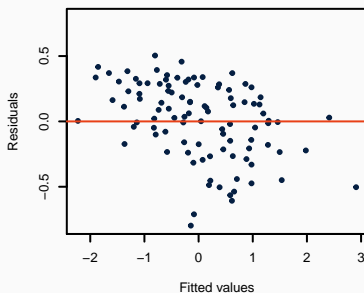
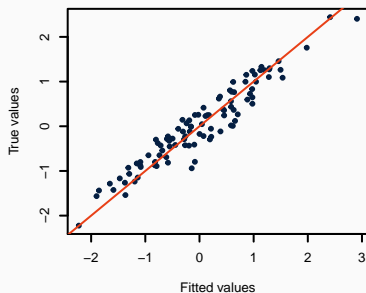
- ▶ $\widehat{C}_{\mathbf{y}, \mathbf{y}'}$ counts the number of times g outputs \mathbf{y} while it should output \mathbf{y}'
- ▶ transposed conventions have been used

Positive and negative

- ▶ when $\mathcal{Y} = \{-1, 1\}$
- ▶ true positive: $g(\mathbf{X}_i) = \mathbf{Y}_i = 1$
- ▶ false negative: $g(\mathbf{X}_i) = -\mathbf{Y}_i = -1$
- ▶ etc.

Diagnostic plots

- ▶ when $\mathcal{Y} = \mathbb{R}$
- ▶ originally for linear regression (standard statistical model)
- ▶ some are useful for general regression models:
 - ▶ scatter plot of \mathbf{Y}_i as a function of $g(\mathbf{X}_i)$
 - ▶ scatter plot of $\mathbf{Y}_i - g(\mathbf{X}_i)$ as a function of $g(\mathbf{X}_i)$ (residual plot)



New data

- ▶ assume g is such as $\hat{R}_l(g, \mathcal{D})$ is small
- ▶ what can we expect on a new data set $\hat{R}_l(g, \mathcal{D}')$?
- ▶ **generalization performances**
- ▶ if g is learned on \mathcal{D} and $\hat{R}_l(g, \mathcal{D}) \ll \hat{R}_l(g, \mathcal{D}')$, g is **overfitting**

New data

- ▶ assume g is such as $\hat{R}_l(g, \mathcal{D})$ is small
- ▶ what can we expect on a new data set $\hat{R}_l(g, \mathcal{D}')$?
- ▶ **generalization performances**
- ▶ if g is learned on \mathcal{D} and $\hat{R}_l(g, \mathcal{D}) \ll \hat{R}_l(g, \mathcal{D}')$, g is **overfitting**

Mathematical model

- ▶ stationary behavior: $\mathcal{D} \simeq \mathcal{D}'$
- ▶ hypotheses:
 - ▶ observations are random variables with values in $\mathcal{X} \times \mathcal{Y}$
 - ▶ they are distributed according to a fixed and **unknown** distribution D
 - ▶ observations are independent

Data set revisited

- ▶ $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$
- ▶ $(\mathbf{X}_i, \mathbf{Y}_i) \sim D$
- ▶ $\mathcal{D} \sim D^N$ (product distribution)

Risk of a model

- ▶ The **risk** of g for the loss function l is

$$R_l(g) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim D}(l(g(\mathbf{X}), \mathbf{Y}))$$

- ▶ we should write $R_l(g, D)$
- ▶ the empirical risk $\hat{R}_l(g, \mathcal{D})$ is a random variable
- ▶ if g is fixed or independent from \mathcal{D} , then $\hat{R}_l(g, \mathcal{D}) \xrightarrow[|\mathcal{D}| \rightarrow \infty]{\text{a.s.}} R_l(g)$
(strong law of large numbers)

Supervised learning

- ▶ input: a data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$ with $\mathcal{D} \sim D^N$ and a loss function l
- ▶ output: a function $g_{\mathcal{D}} : \mathcal{X} \rightarrow \mathcal{Y}$
- ▶ goal: ensure that $R_l(g_{\mathcal{D}})$ is as small as possible
- ▶ **best risk**

$$R_l^* = \inf_{g: \mathcal{X} \rightarrow \mathcal{Y}} R_l(g)$$

Consistency

a machine learning algorithm is universally (i.e. for any D)

- ▶ **consistent** if $\mathbb{E}_{\mathcal{D} \sim D^N}(R_l(g_{\mathcal{D}})) \xrightarrow{N \rightarrow \infty} R_l^*$
- ▶ **strongly consistent** if $R_l(g_{\mathcal{D}}) \xrightarrow[N \rightarrow \infty]{a.s.} R_l^*$

Statistical models

- ▶ specification for D (in general parametric)
- ▶ estimation with maximum likelihood
- ▶ numerous variants (especially Bayesian approaches)

Very different philosophies

- ▶ Machine Learning
 - ▶ performance oriented
 - ▶ universal consistency
 - ▶ limited post learning tuning, exploration, interpretation, etc.
- ▶ Statistical models
 - ▶ strong hypotheses
 - ▶ bad behavior under wrong specification
 - ▶ very rich framework for post estimation exploitation

Statistical models

- ▶ specification for D (in general parametric)
- ▶ estimation with maximum likelihood
- ▶ numerous variants (especially Bayesian approaches)

Very different philosophies

- ▶ Machine Learning
 - ▶ performance oriented
 - ▶ universal consistency
 - ▶ limited post learning tuning, exploration, interpretation, etc.
- ▶ Statistical models
 - ▶ strong hypotheses
 - ▶ bad behavior under wrong specification
 - ▶ very rich framework for post estimation exploitation
- ▶ but many links!

A simple idea

- ▶ $R_I(g)$ cannot be computed as D is unknown
- ▶ but if $g \perp\!\!\!\perp \mathcal{D}$, $\hat{R}_I(g, \mathcal{D}) \xrightarrow[|\mathcal{D}| \rightarrow \infty]{a.s.} R_I(g)$
- ▶ let's replace $R_I(g)$ by $\hat{R}_I(g, \mathcal{D})$!

ERM algorithm

- ▶ choose a class of functions \mathcal{G} from \mathcal{X} to \mathcal{Y}
- ▶ define

$$g_{ERM, \mathcal{D}} = \arg \min_{g \in \mathcal{G}} \hat{R}_I(g, \mathcal{D})$$

Empirical risk can be misleading

- ▶ for the 1-nn, in general:

$$\widehat{R}_l(g_{1-nn}, \mathcal{D}) = 0$$

if g_{1-nn} has been constructed on \mathcal{D}

- ▶ indeed if all the \mathbf{X}_i are distinct, $nn(\mathbf{X}_i, 1) = i$ and thus $g_{1-nn}(\mathbf{X}_i) = \mathbf{Y}_i$
- ▶ unrealistic value for $R_l(g_{1-nn})$

Empirical risk can be misleading

- ▶ for the 1-nn, in general:

$$\widehat{R}_l(g_{1-nn}, \mathcal{D}) = 0$$

if g_{1-nn} has been constructed on \mathcal{D}

- ▶ indeed if all the \mathbf{X}_i are distinct, $nn(\mathbf{X}_i, 1) = i$ and thus $g_{1-nn}(\mathbf{X}_i) = \mathbf{Y}_i$
- ▶ unrealistic value for $R_l(g_{1-nn})$

Source of the problem

- ▶ the strong law of large numbers applies when the random variables are independent
- ▶ the $(l(g_{\mathcal{D}}(\mathbf{X}_i), \mathbf{Y}_i))_{1 \leq i \leq N}$ are dependent variables!

Does ERM work?

- ▶ Yes!

Does ERM work?

- ▶ Yes!
- ▶ but one needs to control the complexity of the class of models \mathcal{G}
- ▶ this is a form of **regularization**: one cannot look for the model in an arbitrary class of models
- ▶ this will be addressed later in the course

Does ERM work?

- ▶ Yes!
- ▶ but one needs to control the complexity of the class of models \mathcal{G}
- ▶ this is a form of **regularization**: one cannot look for the model in an arbitrary class of models
- ▶ this will be addressed later in the course

Basic element for a solution

- ▶ apply the ML method to a data set \mathcal{D} , the learning set
- ▶ evaluate its risk on another independent data set \mathcal{D}'
- ▶ in summary: $\hat{R}_l(g_{\mathcal{D}}, \mathcal{D}')$

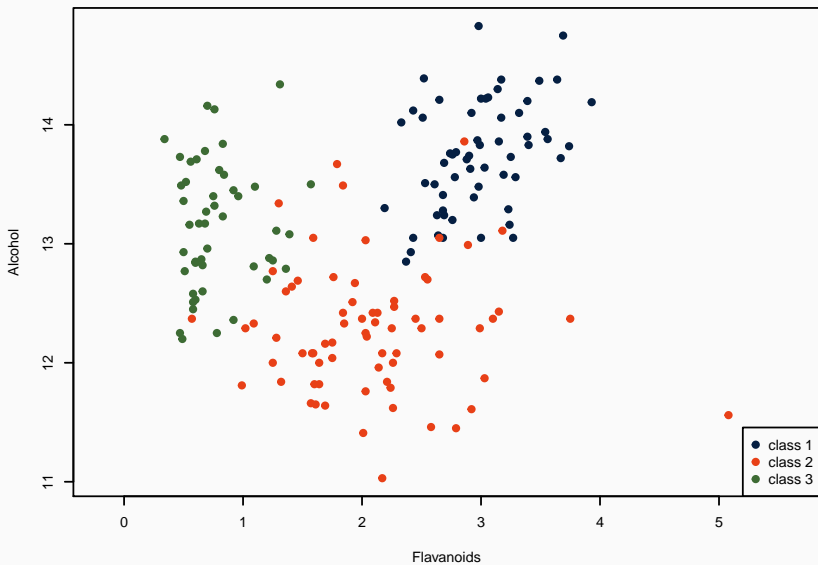
Data and loss

- ▶ chemical analysis of wines derived from three cultivars ($\mathcal{Y} = \{1, 2, 3\}$)
- ▶ 178 observations with 2 variables ($\mathcal{X} = \mathbb{R}^2$)
- ▶ $l(a, b) = \mathbf{1}_{a \neq b}$

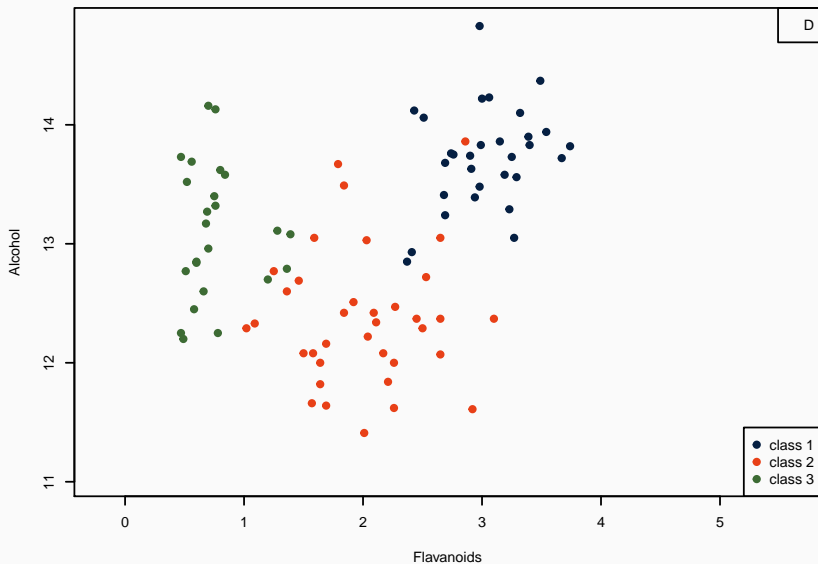
K-nn model

- ▶ use half the data for the learning set \mathcal{D}
- ▶ use the other half for empirical risk evaluation \mathcal{D}'

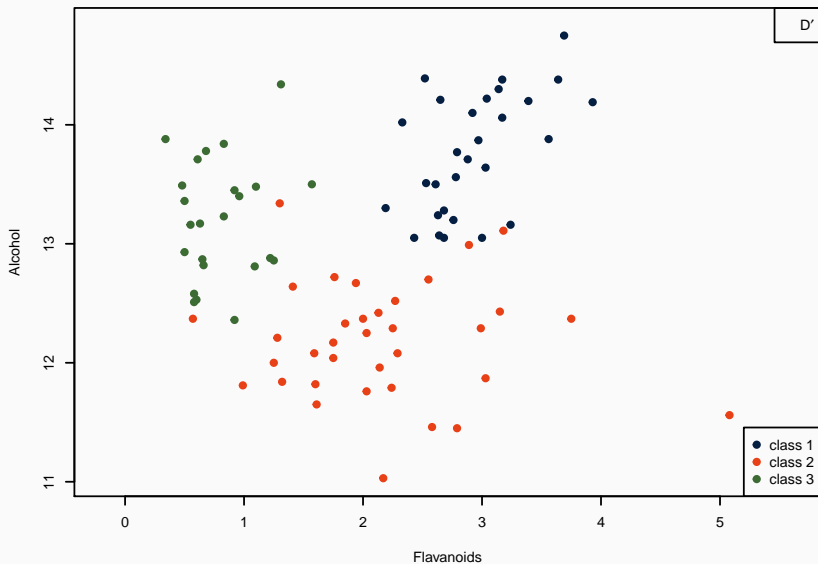
Example



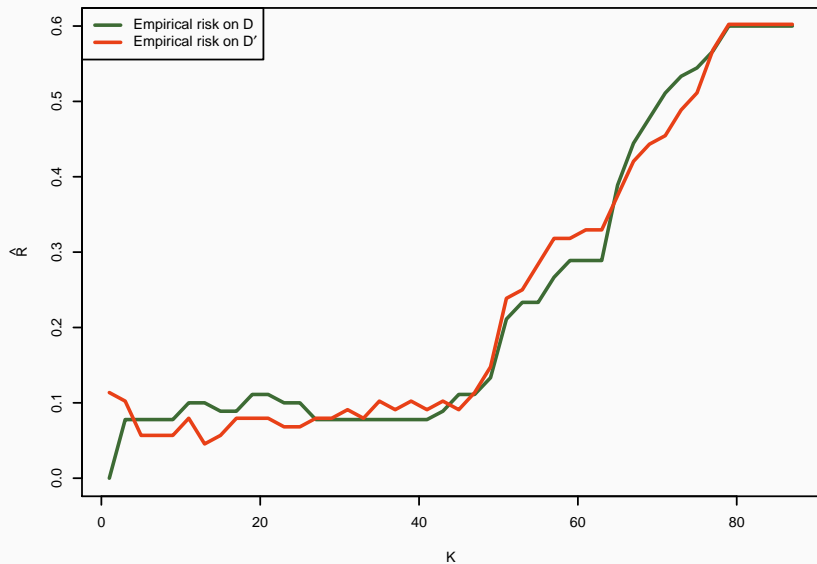
Example



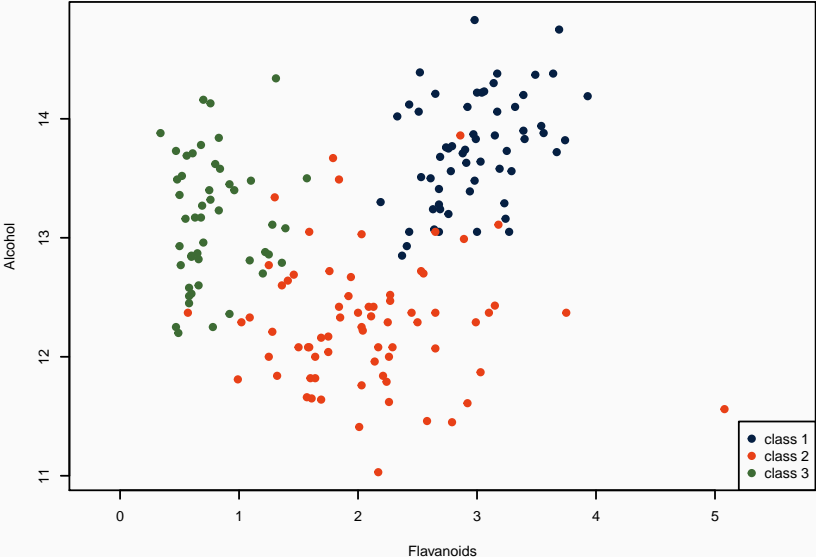
Example



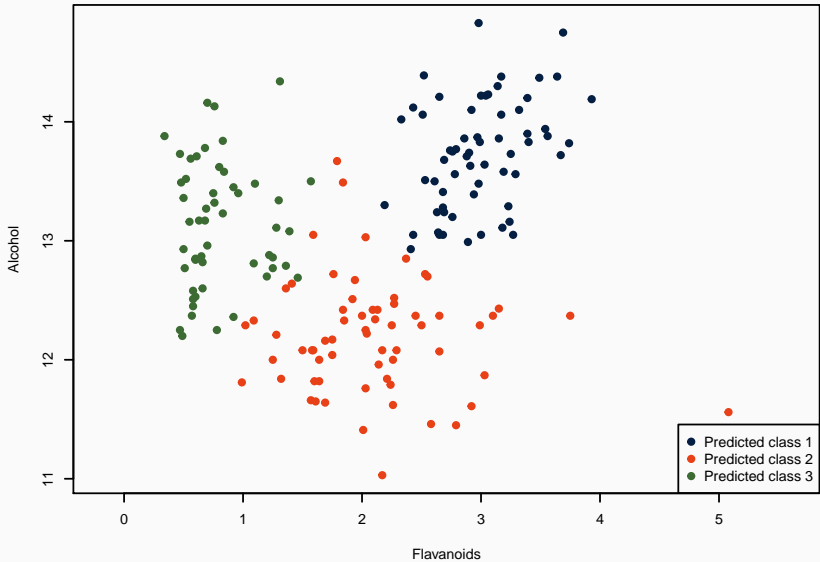
Results



Outputs of the model



Outputs of the model



Basic general framework

1. split the data into \mathcal{D} (learning), \mathcal{D}' (validation) and \mathcal{D}'' (test)
2. for each machine learning algorithm \mathcal{A} under study
 - 2.1 for each value θ of the parameters of the algorithm
 - 2.1.1 compute the model using θ on \mathcal{D} , $g_{\mathcal{A},\theta,\mathcal{D}}$
 - 2.1.2 compute $\widehat{R}_I(g_{\mathcal{A},\theta,\mathcal{D}}, \mathcal{D}')$
3. chose the best model with the best parameter, \mathcal{A}^* and θ^*
(according to $\widehat{R}_I(\cdot, \mathcal{D}')$)
4. compute the best model $g^* = g_{\mathcal{A}^*,\theta^*,\mathcal{D}\cup\mathcal{D}'}$
5. compute $\widehat{R}_I(g^*, \mathcal{D}'')$

Basic general framework

1. split the data into \mathcal{D} (learning), \mathcal{D}' (validation) and \mathcal{D}'' (test)
2. for each machine learning algorithm \mathcal{A} under study
 - 2.1 for each value θ of the parameters of the algorithm
 - 2.1.1 compute the model using θ on \mathcal{D} , $g_{\mathcal{A},\theta,\mathcal{D}}$
 - 2.1.2 compute $\widehat{R}_I(g_{\mathcal{A},\theta,\mathcal{D}}, \mathcal{D}')$
3. chose the best model with the best parameter, \mathcal{A}^* and θ^* (according to $\widehat{R}_I(\cdot, \mathcal{D}')$)
4. compute the best model $g^* = g_{\mathcal{A}^*,\theta^*,\mathcal{D}\cup\mathcal{D}'}$
5. compute $\widehat{R}_I(g^*, \mathcal{D}'')$

Goals of this course

- ▶ describe state-of-the-art alternative for the algorithms
- ▶ study some theoretical aspects, e.g. empirical risk minimization
- ▶ describe better frameworks

Unsupervised learning

- ▶ $\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$
- ▶ no target value!
- ▶ goal: “understanding” the data
- ▶ in practice, many concrete goals such as
 - ▶ finding clusters
 - ▶ finding frequent patterns
 - ▶ finding outliers
 - ▶ modeling the data distribution
 - ▶ etc.

Semi-supervised learning

- ▶ a data set $\mathcal{D} = ((\mathbf{X}_i, \mathbf{Y}_i))_{1 \leq i \leq N}$
- ▶ another data set $\mathcal{D} = ((\mathbf{X}'_i)_{1 \leq i \leq N'})$
- ▶ standard supervised learning goal

Reinforcement learning

- ▶ completely different context:
 - ▶ an **agent** and its **environment** with associated **states**
 - ▶ a set of **actions** the agent can take
 - ▶ **probabilistic transitions**: when the agent takes an action, the global state changes as a consequence, possibly in a stochastic way
 - ▶ **immediate reward**: the reward gained by taking an action
- ▶ goal: computing an optimal **policy**
 - ▶ a policy maps states to actions
 - ▶ the **value** of a policy is the expected total reward obtained by following it
 - ▶ “easy” if everything is known
- ▶ learning: discovering the optimal policy by performing actions