

# Machine Learning and Data Analytics

Eric Medvet    Matilde Trevisani

A.A. 2018/2019

# Section 1

## General information

# Lecturers

- ▶ Matilde Trevisani
  - ▶  $\approx 24$  h (12 CFU only) +  $\approx 24$  h
  - ▶ Dipartimento di Scienze Economiche, Aziendali, Matematiche e Statistiche (DEAMS)
  - ▶ <http://www.units.it/persona/index.php/from/abook/persona/8754>
- ▶ Eric Medvet
  - ▶  $\approx 48$  h
  - ▶ Dipartimento di Ingegneria e Architettura (DIA)
  - ▶ <http://medvet.inginf.units.it/>

# Course materials

- ▶ Lecturer's slides (my part)
  - ▶ <http://medvet.inginf.units.it/teaching/machine-learning-and-data-analytics-2018-2019>
- ▶ Suggested textbooks (for further reading)
  - ▶ Gareth James et al. *An introduction to statistical learning*. Vol. 6. Springer, 2013
  - ▶ Kenneth A De Jong. *Evolutionary computation: a unified approach*. MIT press, 2006
- ▶ Other material:
  - ▶ I'll point you to some scientific papers for discussing examples of application or specific details—just a “chat”

Everything you are required to know is in the lecturer's slides

# How to attend lectures

Everything you are required to know is in the lecturer's slides

**But:** slides are designed assuming that **you** are attending the lecture and **taking notes**

However:

- ▶ lectures will be recorded: <https://videocenter.units.it>
- ▶ during the lectures, I'll (hopely) use the interactive whiteboard for writing annotations
- ▶ including answers to questions posed in the slides, e.g.,
  - ▶ **Q:** is this working?
- ▶ the annotated slides will be available on my website

# How to attend lectures: lab activities

Focus is on methodology, rather than on theory behind techniques:  
how to tackle a problem with ML?

Practicing (in tackling problems) is crucial → lab activities

- ▶  $\approx$  13h
- ▶ mainly **design**, then implementation
  - ▶ **you** practice
  - ▶ I am available any time during/before/after for advising
    - ▶ there's a tutor: Marco Zullich
  - ▶ in general, there is no **one** solution; you make the solution better or worse while (virtually) presenting it
  - ▶ we'll analyze in depth at least one solution
- ▶ form small (2–4) groups
  - ▶ possibly with different background
  - ▶ peer-tutoring

# Exam

Either:

- ▶ a student project **and** a written test
- ▶ a larger written test

Written test: questions on theory and application with medium- and short-length open answers

Project: design, develop, and assess an ML system, choosing among a few options (see <http://medvet.inginf.units.it/teaching/machine-learning-and-data-analytics-2018-2019/student-project>)

# You?

Who are you?



## Section 2

### Introduction

# What is Machine Learning?

## Definition

**Machine Learning** is the science of getting computer to learn without being explicitly programmed.

## Definition

**Data Mining/Analytics** is the science of discovering patterns in data.

# In practice

A set of mathematical and statistical tools for:

- ▶ building a model which allows to predict an output, given an input (*supervised learning*)
  - ▶ example  $\langle \text{input}, \text{output} \rangle$  pairs are available
- ▶ learn relationships and structures in data (*unsupervised learning*)

# Machine Learning everyday

## Example problem: spam

Discriminate between spam and non-spam emails.

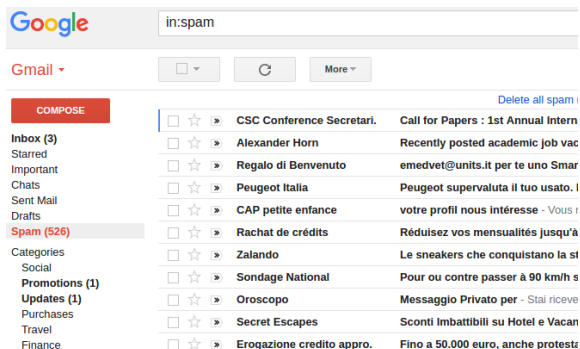


Figure: Spam filtering in Gmail.

# Machine Learning everyday

## Example problem: flight trajectories

Do flights over the same pair  $\langle \text{origin}, \text{destination} \rangle$  follow the “same” trajectory? Why?

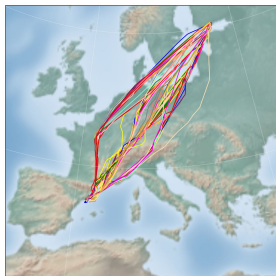


Figure: Clustering of flight trajectories.

# Machine Learning everyday

Example problem: image understanding

Recognize objects in images.

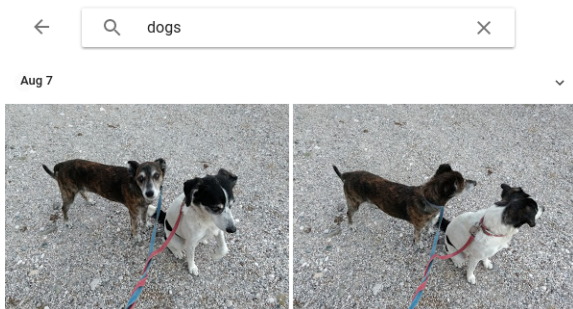
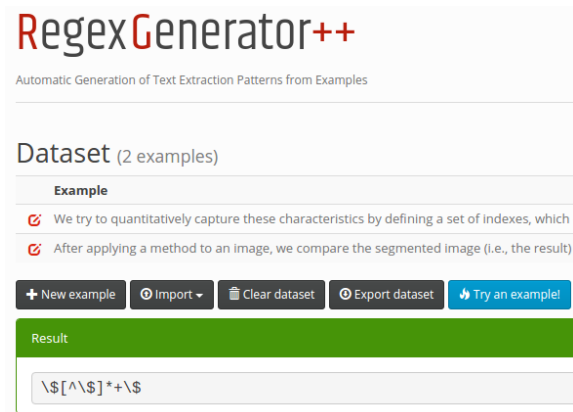


Figure: Object recognition in Google Photos.

# Machine Learning everyday

Example problem: authoring regular expressions

Write a regular expression from matching examples.



The screenshot shows the 'RegexGenerator++' web application. The title is 'RegexGenerator++' in a large, bold font, with 'Regex' in red and 'Generator++' in black. Below the title is the subtitle 'Automatic Generation of Text Extraction Patterns from Examples'. The main section is titled 'Dataset (2 examples)'. It contains a table with two examples. The first example is 'We try to quantitatively capture these characteristics by defining a set of indexes, which c'. The second example is 'After applying a method to an image, we compare the segmented image (i.e., the result) i'. Below the table are five buttons: '+ New example', 'Import', 'Clear dataset', 'Export dataset', and 'Try an example!'. Below the buttons is a green bar labeled 'Result'. Below the green bar is a text input field containing the regular expression '\\$[\^\\\$]\*+\\\$'.

RegexGenerator++

Automatic Generation of Text Extraction Patterns from Examples

Dataset (2 examples)

Example
🔗 We try to quantitatively capture these characteristics by defining a set of indexes, which c
🔗 After applying a method to an image, we compare the segmented image (i.e., the result) i

+ New example Import Clear dataset Export dataset Try an example!

Result

\\$[\^\\\$]\*+\\\$

Figure: Regex generation with <http://regex.inginf.units.it/>.

# Machine Learning everyday

**Q:** what type of learning (supervised/unsupervised) is in the examples?

- ▶ spam
- ▶ image understanding
- ▶ flight trajectories
- ▶ authoring regular expressions



# Why ML/DM “today”?

- ▶ we collect more and more data (*big data*)
- ▶ we have more and more computational power

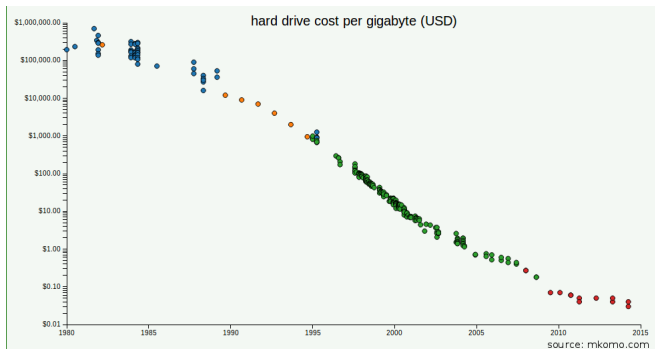


Figure: From <http://www.mkomo.com/cost-per-gigabyte-update>.

# ML/DM is popular!

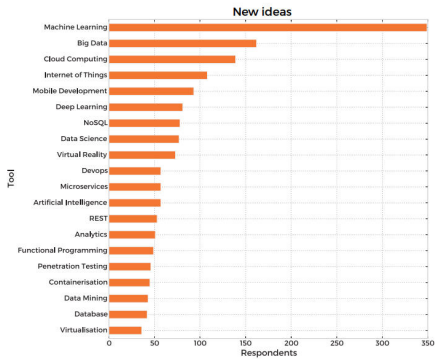


Figure: Popular areas of interest, from the Skill Up 2016: Developer Skills Report<sup>2</sup>

<sup>1</sup><https://techcus.com/p/r1zSmbXut/top-5-highest-paying-programming-languages-of-2016/>.

<sup>2</sup><https://techcus.com/p/r1zSmbXut/top-5-highest-paying-programming-languages-of-2016/>.

# Aims of the course

Be able to:

1. design
2. implement
3. assess experimentally

an end-to-end Machine Learning or Data Mining system.

# Aims of the course

Be able to:

1. design
2. implement
3. assess experimentally

an end-to-end Machine Learning or Data Mining system.

- ▶ Which is the problem to be solved? Which are the input and output? Which are the most suitable techniques? How should data be prepared? Does computation time matter?

# Aims of the course

Be able to:

1. design
2. **implement**
3. assess experimentally

an end-to-end Machine Learning or Data Mining system.

- ▶ Which is the problem to be solved? Which are the input and output? Which are the most suitable techniques? How should data be prepared? Does computation time matter?
- ▶ Write some code!

# Aims of the course

Be able to:

1. design
2. implement
3. **assess experimentally**

an end-to-end Machine Learning or Data Mining system.

- ▶ Which is the problem to be solved? Which are the input and output? Which are the most suitable techniques? How should data be prepared? Does computation time matter?
- ▶ Write some code!
- ▶ How to measure solution quality? How to compare solutions? Is my solution general?

# Aims of the course

Be able to:

1. design
2. implement
3. assess experimentally

an end-to-end Machine Learning or Data Mining system.

- ▶ Which is the problem to be solved? Which are the input and output? Which are the most suitable techniques? How should data be prepared? Does computation time matter?
- ▶ Write some code!
- ▶ How to measure solution quality? How to compare solutions? Is my solution general?
  - ▶ Itself: design and implementation

# Aims of the course: communication

Be able to:

1. design
2. implement
3. assess experimentally

an end-to-end Machine Learning or Data Mining system.

And be able to convince the “client” that it is:

- ▶ technically sound
- ▶ economically viable
- ▶ in its larger context



## Subsection 1

### Motivating example

# The amateur botanist friend

He likes to collect Iris plants. He “realized” that there are 3 species, in particular, that he likes: *Iris setosa*, *Iris virginica*, and *Iris versicolor*. He'd like to have a tool to automatically *classify* collected samples in one of the 3 species.



Figure: Iris versicolor.

How to help him?

# Let's help him

- ▶ Which is the problem to be solved?

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.
  - ▶ Input: the plant sample. . .

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.
  - ▶ Input: the plant sample. . .
    - ▶ a description in natural language?



# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.
  - ▶ Input: the plant sample. . .
    - ▶ a description in natural language?
    - ▶ a digital photo?

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.
  - ▶ Input: the plant sample. . .
    - ▶ a description in natural language?
    - ▶ a digital photo?
    - ▶ DNA sequences?

# Let's help him

- ▶ Which is the problem to be solved?
  - ▶ Assign exactly one specie to a sample.
- ▶ Which are the input and output?
  - ▶ Output: one species among *I. setosa*, *I. virginica*, *I. versicolor*.
  - ▶ Input: the plant sample. . .
    - ▶ a description in natural language?
    - ▶ a digital photo?
    - ▶ DNA sequences?
    - ▶ some measurements of the sample!

## Iris: input and output

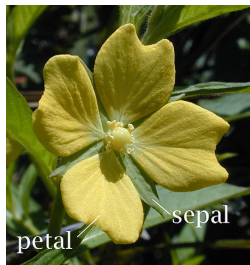


Figure: Sepal and petal.

Input: sepal length and width, petal length and width (in cm)

Output: the class

Example: (5.1, 3.5, 1.4, 0.2)  $\rightarrow$  I. setosa

## Other information

The botanist friend asked a senior botanist to inspect several samples and **label** them with the corresponding species.

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	I. setosa
4.9	3.0	1.4	0.2	I. setosa
7.0	3.2	4.7	1.4	I. versicolor
6.0	2.2	5.0	1.5	I. virginica

# Notation and terminology

- ▶ Sepal length, sepal width, petal length, and petal width are **input variables** (or independent variables, or features, or attributes).
- ▶ Species is the **output variable** (or dependent variable, or response).

# Notation and terminology

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- ▶  $x_1^T = (x_{1,1}, x_{1,2}, \dots, x_{1,p})$  is an **observation** (or instance, or data point), composed of  $p$  variable values;

# Notation and terminology

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- $x_1^T = (x_{1,1}, x_{1,2}, \dots, x_{1,p})$  is an **observation** (or instance, or data point), composed of  $p$  variable values;  $y_1$  is the corresponding output variable value



# Notation and terminology

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & \color{red}{x_{1,2}} & \cdots & x_{1,p} \\ x_{2,1} & \color{red}{x_{2,2}} & \cdots & x_{2,p} \\ \vdots & \color{red}{\vdots} & \ddots & \vdots \\ x_{n,1} & \color{red}{x_{n,2}} & \cdots & x_{n,p} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- ▶  $\mathbf{x}_1^T = (x_{1,1}, x_{1,2}, \dots, x_{1,p})$  is an **observation** (or instance, or data point), composed of  $p$  variable values;  $y_1$  is the corresponding output variable value
- ▶  $\mathbf{x}_2^T = (x_{1,2}, x_{2,2}, \dots, x_{n,2})$  is the vector of all the  $n$  values for the 2nd variable ( $X_2$ ).

# Notation and terminology

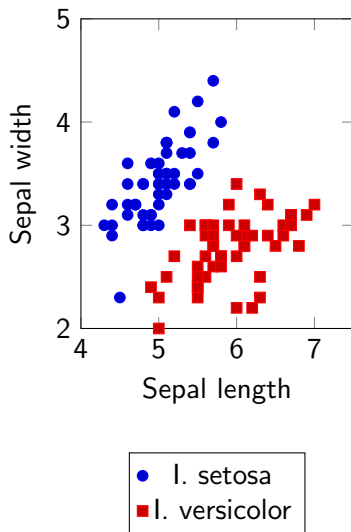
Different communities (e.g., statistical learning vs. machine learning vs. artificial intelligence) use different terms and notation:

- ▶  $x_j^{(i)}$  instead of  $x_{i,j}$  (hence  $x^{(i)}$  instead of  $x_i$ )
- ▶  $m$  instead of  $n$  and  $n$  instead of  $p$
- ▶ ...

Focus on the meaning!

## Iris: visual interpretation

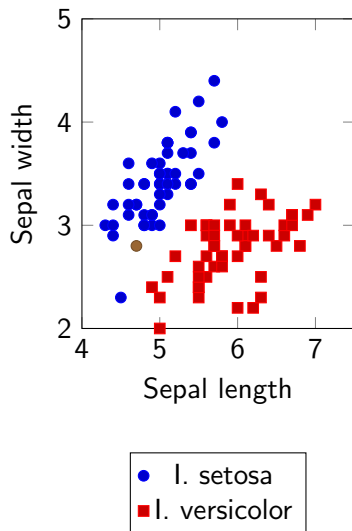
Simplification: forget petal and  
I. virginica  $\rightarrow$  2 variables, 2  
species (**binary classification**  
problem).



# Iris: visual interpretation

Simplification: forget petal and l. virginica  $\rightarrow$  2 variables, 2 species (**binary classification** problem).

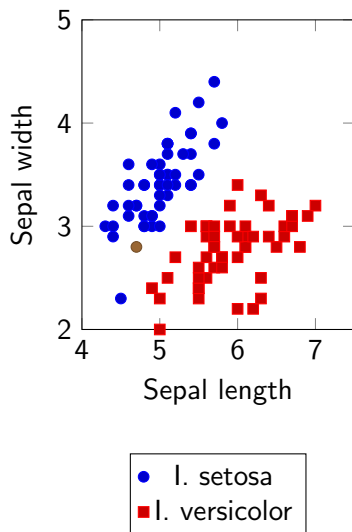
- *Problem:* given any new observation, we want to automatically assign the species.



# Iris: visual interpretation

Simplification: forget petal and  
I. virginica  $\rightarrow$  2 variables, 2  
species (**binary classification**  
problem).

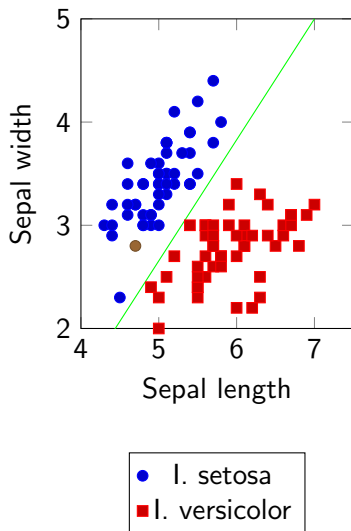
- *Problem:* given any new observation, we want to automatically assign the species.
- *Sketch of a possible solution:*



# Iris: visual interpretation

Simplification: forget petal and l. virginica  $\rightarrow$  2 variables, 2 species (**binary classification** problem).

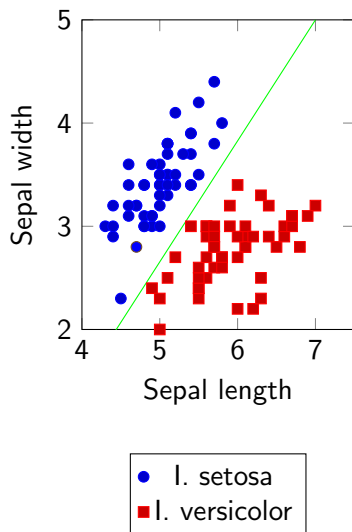
- *Problem*: given any new observation, we want to automatically assign the species.
- *Sketch of a possible solution*:
  1. learn a model (**classifier**)



# Iris: visual interpretation

Simplification: forget petal and I. virginica → 2 variables, 2 species (**binary classification** problem).

- ▶ *Problem*: given any new observation, we want to automatically assign the species.
- ▶ *Sketch of a possible solution*:
  1. learn a model (**classifier**)
  2. “use” model on new observations



# “A” model?

There could be many possible models:

- ▶ how to choose?
- ▶ how to compare?

**Q:** a model of what?



# Choosing the model

The choice of the model/tool/technique to be used is determined by many factors:

- ▶ Problem size ( $n$  and  $p$ )
- ▶ Availability of an output variable ( $y$ )
- ▶ Computational effort (when learning or “using”)
- ▶ Explicability of the model
- ▶ ...

We will see some options.

# Comparing many models

Experimentally: does the model work well on (new) data?

# Comparing many models

Experimentally: does the model work well on (new) data?

Define “works well”:

- ▶ a single performance index?
- ▶ how to measure?
- ▶ repeatability/reproducibility. . .
  - ▶ **Q:** what's the difference?

We will see/discuss some options.

# It does not work well. . .

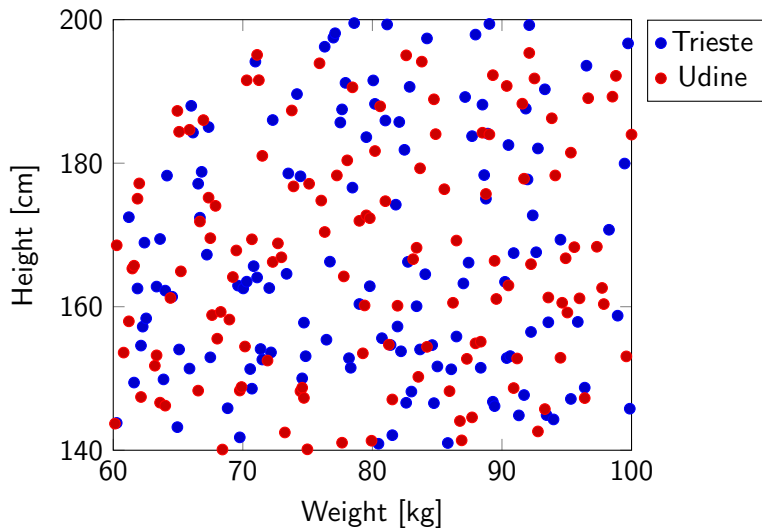
Why?

- ▶ the data is not informative
- ▶ the data is not representative
- ▶ the data has changed
- ▶ the data is too noisy

We will see/discuss these issues.

# ML is not magic

*Problem:* find birth town from height/weight.



**Q:** which is the data issue here?

# Implementation

When “solving” a problem, we usually need:

- ▶ explore/visualize data
- ▶ apply one or more ML technique
- ▶ assess learned models

“By hands?” No, with software!

# ML/DM software

Many options:

- ▶ libraries for general purpose languages:
  - ▶ Java: e.g., <http://haifengl.github.io/smile/>
  - ▶ Python: e.g., <http://scikit-learn.org/stable/>
  - ▶ ...
- ▶ specialized sw environments:
  - ▶ Octave: [https://en.wikipedia.org/wiki/GNU\\_Octave](https://en.wikipedia.org/wiki/GNU_Octave)
  - ▶ R: [https://en.wikipedia.org/wiki/R\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/R_(programming_language))
- ▶ from scratch

# ML/DM software: which one?

- ▶ production/prototype
- ▶ platform constraints
- ▶ degree of (data) customization
- ▶ documentation availability/community size
- ▶ ...
- ▶ previous knowledge/skills



# ML/DM software: why?

In all cases, sw allows to be more productive and concise.

E.g., learn and use a model for classification, in Java+Smile:

```
1 double[] [] instances = ...;
2 int[] labels = ...;
3 RandomForest classifier = (new RandomForest.Trainer()).train(
    instances, labels);
4 double[] newInstance = ...;
5 int newLabel = classifier.predict(newInstance);
```

In R:

```
1 d = ...
2 classifier = randomForest(label~., d)
3 newD = ...
4 newLabels = predict(classifier, newD)
```

We will work with R.

## Section 3

# Fundamentals of R

# R software

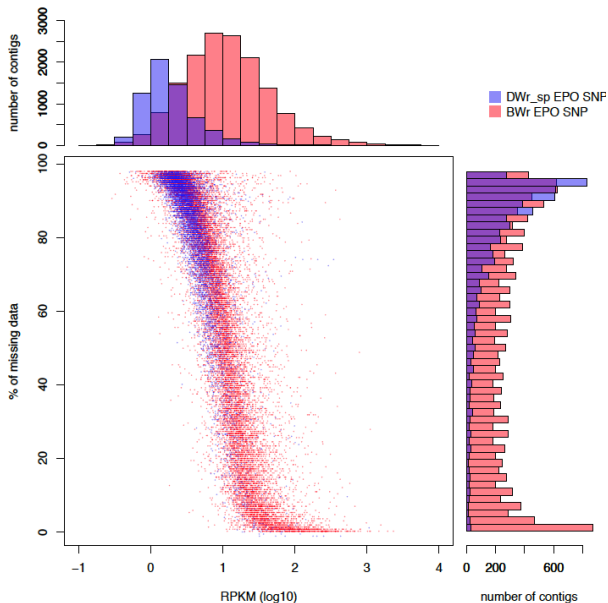
- ▶ R
  - ▶ a programming language
  - ▶ a software environment
- ▶ RStudio
  - ▶ an IDE built on R

## Section 4

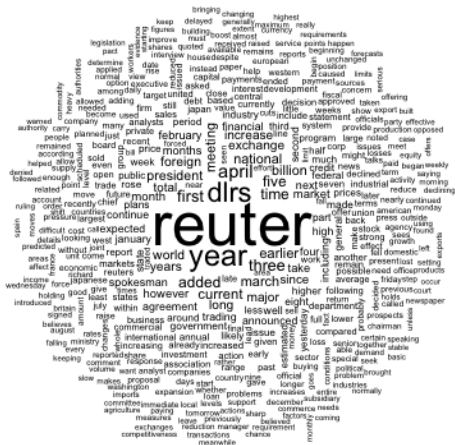
### Plotting data: an overview



# Aim of a plot: examples

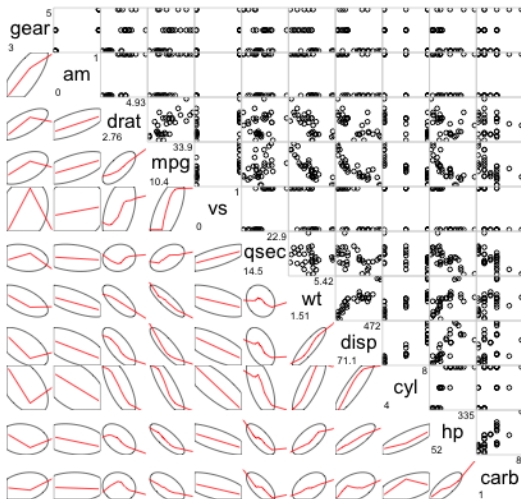


## Aim of a plot: examples



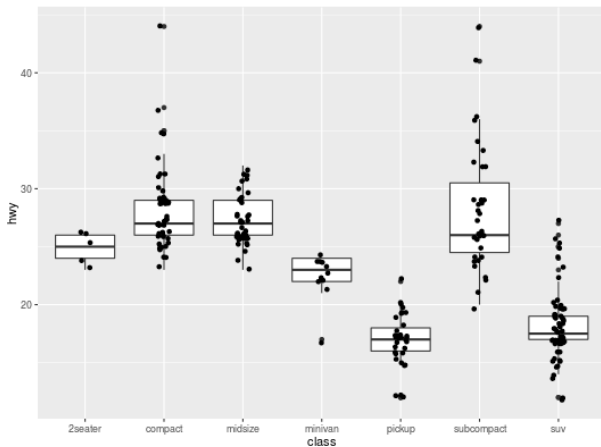
# Aim of a plot: examples

Car Milage Data in PC2/PC1 Order





# Aim of a plot: examples



# Lab: let's know iris (1 h)

1. get iris data
2. know basic info about it (`summary`)
3. plot iris and play with it

Hints:

```
dr = iris %>% group_by(Species) %>% summarise(Avg.Sepal.Ratio=  
mean(Sepal.Length/Sepal.Width),Avg.Petal.Ratio=mean(Petal.Length/  
Petal.Width))
```

```
dr %>% gather(ratio, value, -Species)
```

Packages: `ggplot2`, `dplyr`, `tidyr`

## Section 5

### Tree-based methods

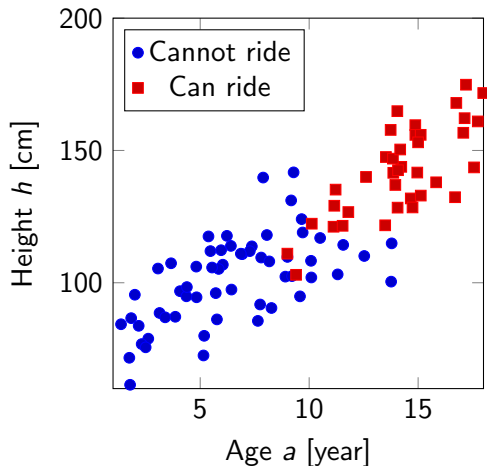
# The carousel robot attendant

*Problem:* replace the carousel attendant with a robot which automatically decides who can ride the carousel.



## Carousel: data

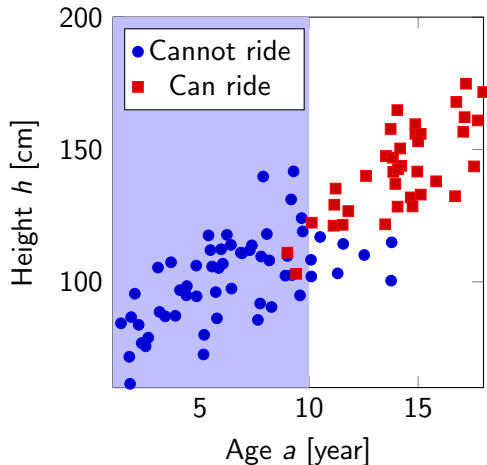
Observed human attendant's decisions.



How can the robot take the decision?

## Carousel: data

Observed human attendant's decisions.

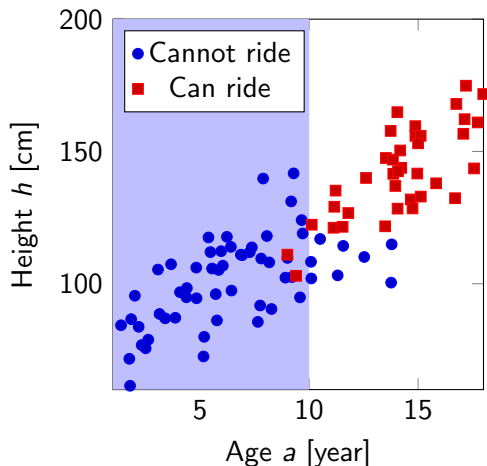


How can the robot take the decision?

- ▶ if younger than 10  $\rightarrow$  can't!

## Carousel: data

Observed human attendant's decisions.

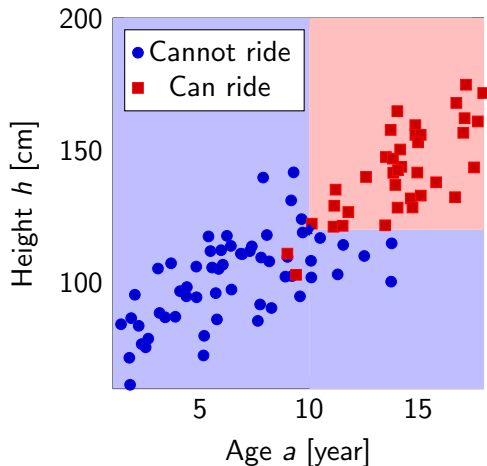


How can the robot take the decision?

- ▶ if younger than 10  $\rightarrow$  can't!
- ▶ otherwise:

## Carousel: data

Observed human attendant's decisions.



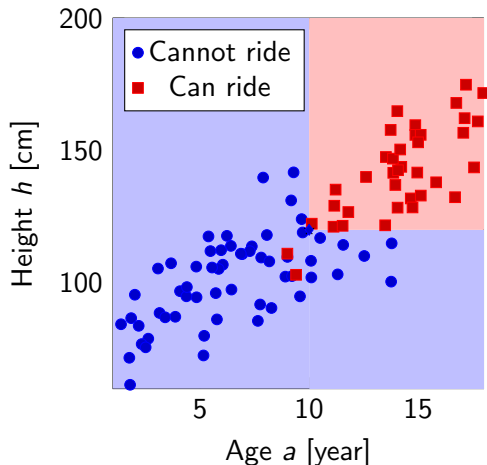
How can the robot take the decision?

- ▶ if younger than 10  $\rightarrow$  can't!
- ▶ otherwise:
  - ▶ if shorter than 120  $\rightarrow$  can't!
  - ▶ otherwise  $\rightarrow$  can!



# Carousel: data

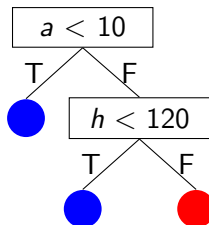
Observed human attendant's decisions.



How can the robot take the decision?

- ▶ if younger than 10  $\rightarrow$  can't!
- ▶ otherwise:
  - ▶ if shorter than 120  $\rightarrow$  can't!
  - ▶ otherwise  $\rightarrow$  can!

Decision tree!



# How to build a decision tree

Dividi-et-impera (recursively):

- ▶ find a cut variable and a cut value
- ▶ for left-branch, dividi-et-impera
- ▶ for right-branch, dividi-et-impera

# How to build a decision tree: detail

Recursive binary splitting

```
function BUILDDECISIONTREE( $\mathbf{X}, \mathbf{y}$ )  
  if SHOULDSTOP( $\mathbf{y}$ ) then  
     $\hat{y} \leftarrow$  most common class in  $\mathbf{y}$   
    return new terminal node with  $\hat{y}$   
  else  
     $(i, t) \leftarrow$  BESTBRANCH( $\mathbf{X}, \mathbf{y}$ )  
     $n \leftarrow$  new branch node with  $(i, t)$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t}$ ) to  $n$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t}$ ) to  $n$   
    return  $n$   
  end if  
end function
```

- ▶ Recursive binary splitting
- ▶ Top down (start from the “big” problem)

## Best branch

```
function BESTBRANCH(X, y)  
     $(i^*, t^*) \leftarrow \arg \min_{i,t} E(\mathbf{y}|_{\mathbf{x}_i \geq t}) + E(\mathbf{y}|_{\mathbf{x}_i < t})$   
    return  $(i^*, t^*)$   
end function
```

Classification error on subset:

$$E(\mathbf{y}) = \frac{|\{y \in \mathbf{y} : y \neq \hat{y}\}|}{|\mathbf{y}|}$$

$\hat{y}$  = the most common class in  $\mathbf{y}$

- Greedy (choose split to minimize error now, not in later steps)

## Best branch

$$(i^*, t^*) \leftarrow \arg \min_{i, t} E(\mathbf{y} | \mathbf{x}_i \geq t) + E(\mathbf{y} | \mathbf{x}_i < t)$$

The formula say what is done, not how is done!

**Q:** different “how” can differ? how?

## Stopping criterion

```
function SHOULDSTOP(y)  
  if y contains only one class then  
    return true  
  else if  $|\mathbf{y}| < k_{\min}$  then  
    return true  
  else  
    return false  
  end if  
end function
```

Other possible criterion:

- ▶ tree depth larger than  $d_{\max}$

## Best branch criteria

Classification error  $E()$  works, but has been shown to be “not sufficiently sensitive for tree-growing”.

$$E(\mathbf{y}) = \frac{|\{y \in \mathbf{y} : y \neq \hat{y}\}|}{|\mathbf{y}|} = 1 - \max_c \frac{|\{y \in \mathbf{y} : y = c\}|}{|\mathbf{y}|} = 1 - \max_c p_{\mathbf{y},c}$$

Other two options:

- Gini index

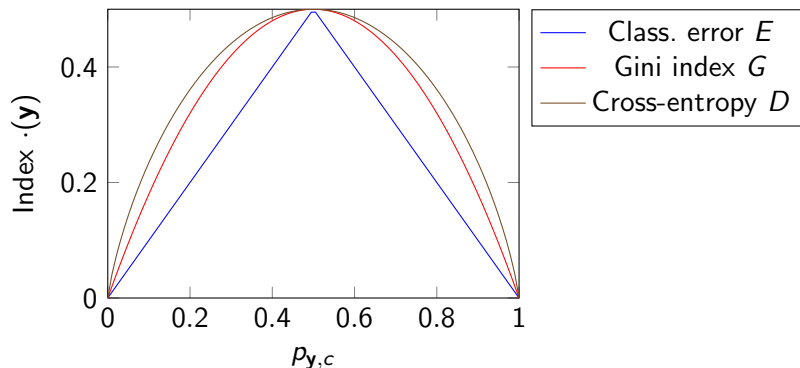
$$G(\mathbf{y}) = \sum_c p_{\mathbf{y},c}(1 - p_{\mathbf{y},c})$$

- Cross-entropy

$$D(\mathbf{y}) = - \sum_c p_{\mathbf{y},c} \log p_{\mathbf{y},c}$$

For all indexes, the lower the better (**node impurity**).

## Best branch criteria: binary classification



Cross-entropy is rescaled.

**Q:** what happens with multiclass problems?

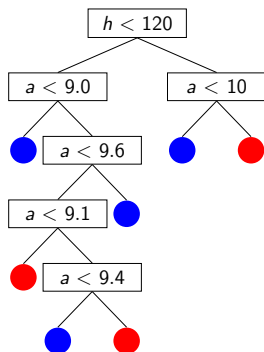
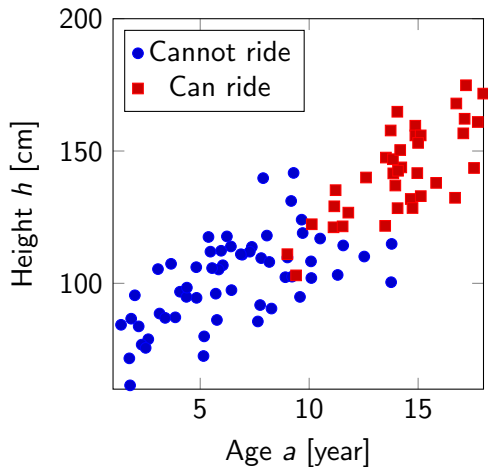


# Categorical independent variables

- ▶ Trees can work with categorical variables
- ▶ Branch node is  $x_i = c$  or  $x_i \in C' \subset C$  ( $c$  is a class)
- ▶ Can mix categorical and numeric variables

# Stopping criterion: role of $k_{\min}$

Suppose  $k_{\min} = 1$  (never stop for  $y$  size)



**Q:** what's wrong? (recall: "a model of what?")

# Tree complexity

When the tree is “too complex”

- ▶ less readable/understandable/explicable
- ▶ maybe there was noise into the data

**Q:** what's noise in carousel data?

Tree complexity is not related (only) with  $k_{\min}$ , but also with data

## Tree complexity: other interpretation

- ▶ maybe there was noise into the data

The tree *fits* the learning data too much:

- ▶ it overfits (**overfitting**)
- ▶ does not generalize (high **variance**: model varies if learning data varies)

# High variance

“model varies if learning data varies”: what? why data varies?

- ▶ learning data is about the system/phenomenon/nature  $S$ 
  - ▶ a collection of *observations* of  $S$
  - ▶ a point of view on  $S$

# High variance

“model varies if learning data varies”: what? why data varies?

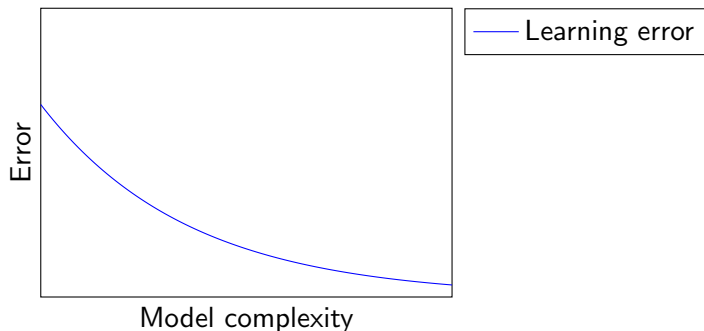
- ▶ learning data is about the system/phenomenon/nature  $S$ 
  - ▶ a collection of *observations* of  $S$
  - ▶ a point of view on  $S$
- ▶ learning is about understanding/knowing/explaining  $S$

# High variance

“model varies if learning data varies”: what? why data varies?

- ▶ learning data is about the system/phenomenon/nature  $S$ 
  - ▶ a collection of *observations* of  $S$
  - ▶ a point of view on  $S$
- ▶ learning is about understanding/knowing/explaining  $S$ 
  - ▶ if I change the point of view on  $S$ , my knowledge about  $S$  **should remain the same!**

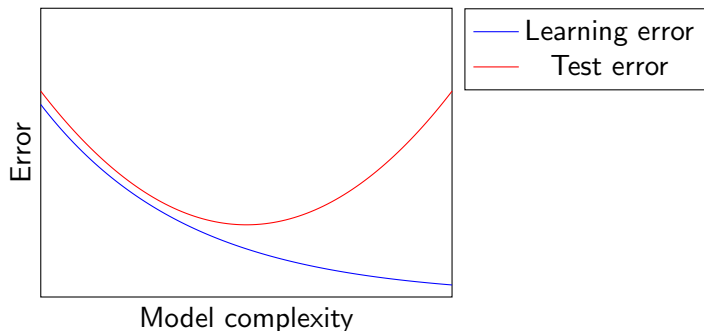
# Spotting overfitting



Test error: error on unseen data



# Spotting overfitting



Test error: error on unseen data

# $k$ -fold cross-validation


























Where can I find “unseen data”? Pretend to have it!

1. split learning data ( $\mathbf{X}$  and  $\mathbf{y}$ ) in  $k$  equal slices (each of  $\frac{n}{k}$  observations/elements)
2. for each split (i.e., each  $i \in \{1, \dots, k\}$  )
  - 2.1 learn on all but  $k$ -th slice
  - 2.2 compute classification error on **unseen**  $k$ -th slice
3. average the  $k$  classification errors

In essence:

- ▶ can the learner generalize beyond available data?
- ▶ how the learned artifact will behave on unseen data?

## k-fold cross-validation

folding 1						error <sub>1</sub>
folding 2						error <sub>2</sub>
folding 3						error <sub>3</sub>
folding 4						error <sub>4</sub>
folding 5						error <sub>5</sub>

$$\text{error} = \frac{1}{k} \sum_{i=1}^{i=k} \text{error}_i$$

Or with any other meaningful (effectiveness) measure

**Q:** how should data be split?

# Fighting overfitting with trees

- ▶ large  $k_{\min}$  (large w.r.t. what?)
- ▶ when building, limit depth
- ▶ when building, don't split if low overall impurity decrease
- ▶ after building, *prune*

# Pruning: high level idea

1. learn a full tree  $t_0$
2. build from  $t_0$  a sequence  $\mathcal{T} = \{t_0, t_1, \dots, t_n\}$  of trees such that
  - ▶  $t_i$  is a root-subtree of  $t_{i-1}$  ( $t_i \subset t_{i-1}$ )
  - ▶  $t_i$  is always less complex than  $t_{i-1}$
3. choose the  $t \in \mathcal{T}$  with minimum classification error with  $k$ -fold cross-validation

# $k$ -fold cross-validation: data splitting

**Q:** how should data be split?

Example: Android Malware detection

- ▶ Gerardo Canfora et al. “Effectiveness of opcode ngrams for detection of multi family android malware”. In: *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE. 2015, pp. 333–340
- ▶ Gerardo Canfora et al. “Detecting android malware using sequences of system calls”. In: *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*. ACM. 2015, pp. 13–20

# Android Malware detection

# Using cross-validation (CV) for assessment (I)

How the learned artifact will behave on unseen data?

More precisely:

How an artifact learned with **this learning technique** will behave on unseen data?



## Using CV for assessment (II)

“This learning technique” = `BUILDDECISIONTREE()` with  $k_{\min} = 10$

1. repeat  $k$  times

- 1.1 `BUILDDECISIONTREE()` with  $k_{\min} = 10$  on all but one slice

- ▶  $\frac{k}{k-1}n$  observations in each  $\mathbf{X}$  passed to `BUILDDECISIONTREE()`

- 1.2 compute classification error on left out slice

2. average computed classification errors

$k$  invocations of `BUILDDECISIONTREE()`

## Using CV for assessment (III)

“This learning technique” = `BUILDDECISIONTREE()` with  $k_{\min}$  chosen automatically with a 10-fold CV

For assessing this technique, we do two nested CVs:

1. repeat  $k$  times
  - 1.1 choose  $k_{\min}$  among  $m$  values with 10-CV (repeat `BUILDDECISIONTREE()`  $10m$  times) on all but one slice
    - ▶  $\frac{k-1}{k} \frac{9}{10} n$  observations in each  $\mathbf{X}$  passed to `BUILDDECISIONTREE()`!
  - 1.2 compute classification error on left out slice
    - ▶ usually, a new tree is built on  $\frac{k-1}{k} n$  observations
2. average computed classification errors

$(10m + 1)k$  invocations of `BUILDDECISIONTREE()`

## Using CV for assessment: “cheating”

“This learning technique” = `BUILDDECISIONTREE()` with  $k_{\min}$  chosen automatically with a 10-fold CV

Using just one CV is cheating (cherry picking)!

- ▶  $k_{\min}$  is chosen exactly to minimize error on the full dataset
- ▶ conceptually, this way of “fitting”  $k_{\min}$  is similar to the way we build the tree

## Subsection 1

### Regression trees

# Regression with trees

Trees can be used for regression, instead of classification.

decision tree vs. regression tree

## Tree building: decision $\rightarrow$ regression

```
function BUILDDECISIONTREE( $\mathbf{X}, \mathbf{y}$ )  
  if SHOULDSTOP( $\mathbf{y}$ ) then  
     $\hat{y} \leftarrow$  most common class in  $\mathbf{y}$   
    return new terminal node with  $\hat{y}$   
  else  
     $(i, t) \leftarrow$  BESTBRANCH( $\mathbf{X}, \mathbf{y}$ )  
     $n \leftarrow$  new branch node with  $(i, t)$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t}$ ) to  $n$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t}$ ) to  $n$   
    return  $n$   
  end if  
end function
```

**Q:** what should we change?

## Tree building: decision $\rightarrow$ regression

```
function BUILDDECISIONTREE( $\mathbf{X}, \mathbf{y}$ )  
  if SHOULDSTOP( $\mathbf{y}$ ) then  
     $\hat{y} \leftarrow \bar{y}$  ▷ mean  $\mathbf{y}$   
    return new terminal node with  $\hat{y}$   
  else  
     $(i, t) \leftarrow \text{BESTBRANCH}(\mathbf{X}, \mathbf{y})$   
     $n \leftarrow$  new branch node with  $(i, t)$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i < t}, \mathbf{y}|_{\mathbf{x}_i < t}$ ) to  $n$   
    append child BUILDDECISIONTREE( $\mathbf{X}|_{\mathbf{x}_i \geq t}, \mathbf{y}|_{\mathbf{x}_i \geq t}$ ) to  $n$   
    return  $n$   
  end if  
end function
```

**Q:** what should we change?

## Best branch

```
function BESTBRANCH(X, y)  
     $(i^*, t^*) \leftarrow \arg \min_{i,t} E(\mathbf{y} | \mathbf{x}_i \geq t) + E(\mathbf{y} | \mathbf{x}_i < t)$   
    return  $(i^*, t^*)$   
end function
```

**Q:** what should we change?



## Best branch

**function** BESTBRANCH(**X**, **y**)

$(i^*, t^*) \leftarrow \arg \min_{i,t} \sum_{y_i \in \mathbf{y} | x_i \geq t} (y_i - \bar{y})^2 + \sum_{y_i \in \mathbf{y} | x_i < t} (y_i - \bar{y})^2$

**return**  $(i^*, t^*)$

**end function**

**Q:** what should we change?

Minimize sum of residual sum of squares (RSS) (the two  $\bar{y}$  are different)

## Stopping criterion

```
function SHOULDSTOP(y)  
  if y contains only one class then  
    return true  
  else if  $|\mathbf{y}| < k_{\min}$  then  
    return true  
  else  
    return false  
  end if  
end function
```

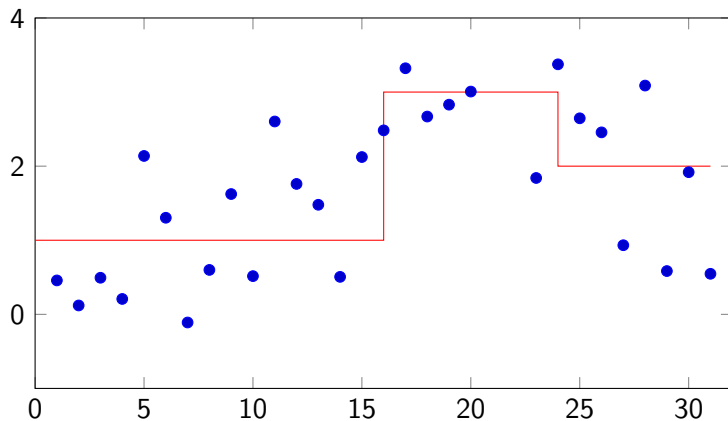
**Q:** what should we change?

## Stopping criterion

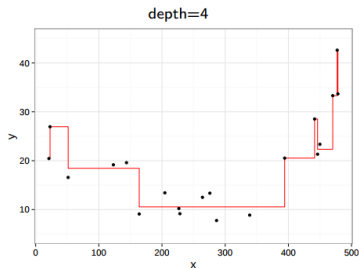
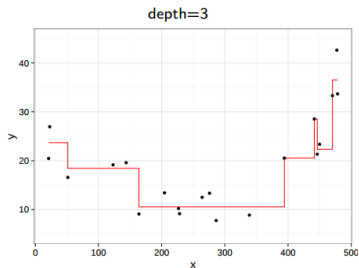
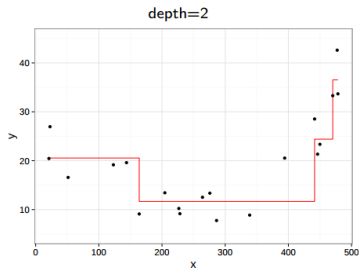
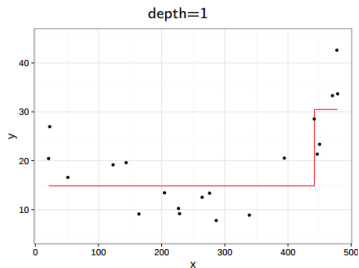
```
function SHOULDSTOP(y)  
  if RSS is 0 then  
    return true  
  else if  $|y| < k_{\min}$  then  
    return true  
  else  
    return false  
  end if  
end function
```

**Q:** what should we change?

# Interpretation



# Regression and overfitting



# Trees in summary

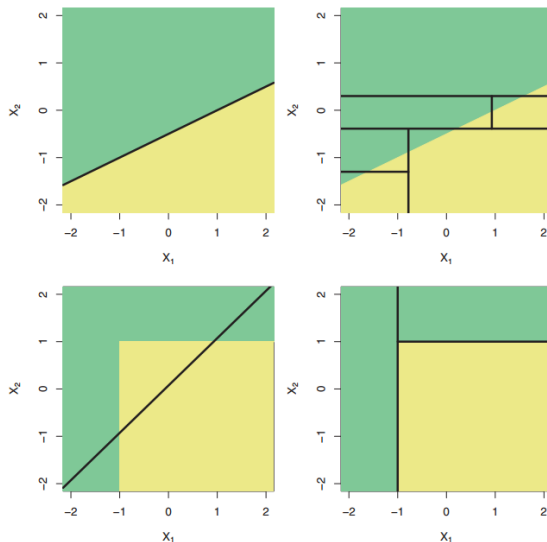
## Pros:

- ▲ easily interpretable/explicable
- ▲ learning and regression/classification easily understandable
- ▲ can handle both numeric and categorical values

## Cons:

- ▼ not so accurate (**Q**: always?)

# Tree accuracy?



## Lab: tree on iris (2 h)

- ▶ for each of the 5 variables in iris, predict it with the other 4
- ▶ which is the hardest to be predicted? why?

Packages: `tree`

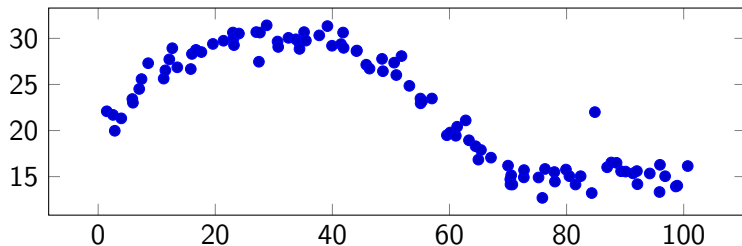
Functions: `tree`, `prune.tree`, `predict.tree(t, type="class")`, `table`



## Subsection 2

### Trees aggregation

# Weakness of the tree



Small tree:

- ▶ low complexity
- ▶ will hardly fit the “curve” part
- ▶ *high bias, low variance*

Big tree:

- ▶ high complexity
- ▶ may overfit the noise on the right part
- ▶ *low bias, high variance*



# Big tree view

A big tree:

- ▶ has a detailed view of the learning data (high complexity)
- ▶ “trusts too much” the learning data (high variance)

What if we “combine” different big tree views and ignore details on which they disagree?

# Wisdom of the crowds

What if we “combine” different big tree views and ignore details on which they disagree?

- ▶ many views
- ▶ independent views
- ▶ aggregation of views

$\approx$  *the wisdom of the crowds*: a collective opinion may be better than a single expert's opinion

# Wisdom of the trees

- ▶ many views
- ▶ independent views
- ▶ aggregation of views

# Wisdom of the trees

- ▶ many views
  - ▶ just use many trees
- ▶ independent views
- ▶ aggregation of views

# Wisdom of the trees

- ▶ many views
  - ▶ just use many trees
- ▶ independent views
- ▶ aggregation of views
  - ▶ just average prediction (regression) or take most common prediction (classification)



# Wisdom of the trees

- ▶ many views
  - ▶ just use many trees
- ▶ independent views
  - ▶ ??? learning is deterministic: same data  $\Rightarrow$  same tree  $\Rightarrow$  same view
- ▶ aggregation of views
  - ▶ just average prediction (regression) or take most common prediction (classification)

# Independent views

Independent views  $\equiv$  different points of view  $\equiv$  *different* learning data

But we have only *one* learning data!

# Independent views: idea! (**Bootstrap**)

Like in cross-fold, consider only a part of the data, but:

- ▶ instead of a subset
- ▶ a sample with repetitions

# Independent views: idea! (**Bootstrap**)

Like in cross-fold, consider only a part of the data, but:

- ▶ instead of a subset
- ▶ a sample with repetitions

$\mathbf{X} = (x_1^T x_2^T x_3^T x_4^T x_5^T)$	original learning data
$\mathbf{X}_1 = (x_1^T x_5^T x_3^T x_2^T x_5^T)$	sample 1
$\mathbf{X}_2 = (x_4^T x_2^T x_3^T x_1^T x_1^T)$	sample 2
$\mathbf{X}_i = \dots$	sample $i$

- ▶ ( $\mathbf{y}$  omitted for brevity)
- ▶ learning data size is not a limitation (differently than with subset)

# Tree bagging

When learning:

1. Repeat  $B$  times
  - 1.1 take a sample of the learning data
  - 1.2 learn a tree (unpruned)

When predicting:

1. Repeat  $B$  times
  - 1.1 get a prediction from  $i$ th learned tree
2. predict the average (or most common) prediction

For classification, other aggregations can be done: majority voting (most common) is the simplest

Using independent, possibly different classifiers together: *ensemble* of classifiers

# How many trees?

$B$  is a parameter:

- ▶ when there is a parameter, there is the problem of finding a good value
- ▶ remember  $k_{\min}$ , depth (Q: impact on?)

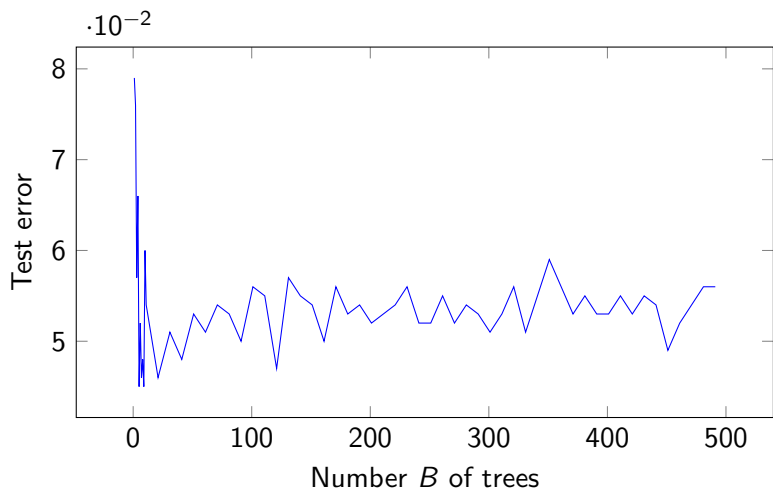
# How many trees?

$B$  is a parameter:

- ▶ when there is a parameter, there is the problem of finding a good value
- ▶ remember  $k_{\min}$ , depth (Q: impact on?)
- ▶ it has been shown (experimentally) that
  - ▶ for “large”  $B$ , bagging is better than single tree
  - ▶ increasing  $B$  does not cause overfitting
  - ▶ (for us: default  $B$  is ok! “large”  $\approx$  hundreds)

Q: how better? at which cost?

## Bagging: impact of $B$





## Independent view: improvement

Despite being learned on different samples, bagging trees may be correlated, hence views are not very independent

- ▶ e.g., one variable is much more important than others for predicting (*strong predictor*)

Idea: force point of view differentiation by “hiding” variables

# Random forest

When learning:

1. Repeat  $B$  times
  - 1.1 take a sample of the learning data
  - 1.2 consider only  $m$  on  $p$  independent variables
  - 1.3 learn a tree (unpruned)

When predicting:

1. Repeat  $B$  times
  - 1.1 get a prediction from  $i$ th learned tree
2. predict the average (or most common) prediction

- ▶ (observations and) variables are **randomly** chosen. . .
- ▶ . . . to learn a **forest** of trees

**Q:** are missing variables a problem?

# Random forest: parameter $m$

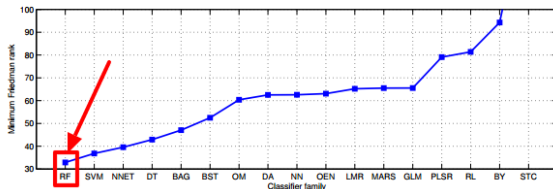
How to choose the value for  $m$ ?

- ▶  $m = p \rightarrow$  bagging
- ▶ it has been shown (experimentally) that
  - ▶  $m$  does not relate with overfitting
  - ▶  $m = \sqrt{p}$  is good for classification
  - ▶  $m = \frac{p}{3}$  is good for regression
  - ▶ (for us, default  $m$  is ok!)

# Random forest

Experimentally shown: one of the “best” multi-purpose supervised classification methods

- Manuel Fernández-Delgado et al. “Do we need hundreds of classifiers to solve real world classification problems”. In: *J. Mach. Learn. Res* 15.1 (2014), pp. 3133–3181



but...

# No free lunch!

“Any two optimization algorithms are equivalent when their performance is averaged across all possible problems”

- ▶ David H Wolpert. “The lack of a priori distinctions between learning algorithms”. In: *Neural computation* 8.7 (1996), pp. 1341–1390

Why free lunch?

- ▶ many restaurants, many items on menus, many possibly prices for each item: where to go to eat?
- ▶ no general answer
- ▶ but, if you are a vegan, or like pizza, then a best choice could exist

**Q:** problem? algorithm?

# Observation sampling

When learning:

1. Repeat  $B$  times
  - 1.1 take a sample of the learning data
  - 1.2 consider only  $m$  on  $p$  independent variables (only for RF)
  - 1.3 learn a tree (unpruned)

Each learned tree uses only a portion of the observation in the learning data:

- ▶ for each observation,  $\approx \frac{B}{3}$  trees did not consider it when learned

# Observation sampling

When learning:

1. Repeat  $B$  times
  - 1.1 take a sample of the learning data
  - 1.2 consider only  $m$  on  $p$  independent variables (only for RF)
  - 1.3 learn a tree (unpruned)

Each learned tree uses only a portion of the observation in the learning data:

- ▶ for each observation,  $\approx \frac{B}{3}$  trees did not consider it when learned
- ▶ those observation were *unseen* for those trees, like in cross-validation (**OOB = out-of-bag**)

## Bonus 1: OOB error

- ▶ for unseen each observation there are  $\frac{B}{3}$  predictions
- ▶ can “average” prediction among trees, observation and obtain an estimate of the testing error (OOB error)
  - ▶ like with cross-fold validation
  - ▶ for free!



# OOB error

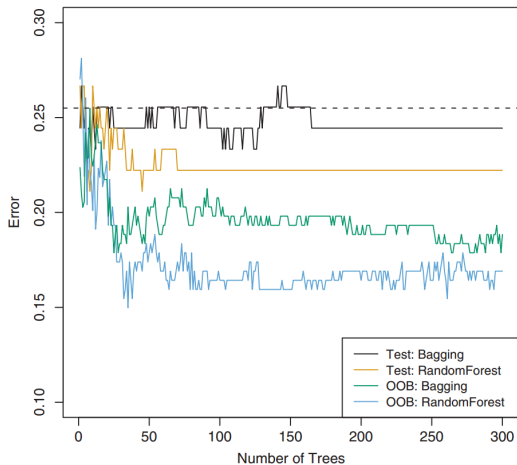


Image from An Introduction to Statistical Learning

# Why estimating the test error?

Because the test data, in real world, is not available!

- ▶ will my ML solution work?

# Bagging/RF and explicability

- ▶ Trees are easily understandable  $\rightarrow$  explicability
- ▶ Hundreds of trees are not!

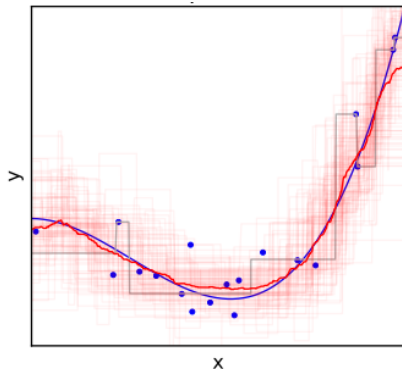


Image from F. Daolio

# Bagging/RF and explicability: idea!

While learning:

1. for each tree, at each split
  - 1.1 keep note of the split variable
  - 1.2 keep note of RSS/Gini reduction
2. for each variable, sum reductions

The largest reduction, the more important the variable!

## Bonus 2: variable importance

Instead of explicability based on tree shape:

- ▶ importance of variables based on RSS/Gini reduction

# Nature of the prediction

Consider classification:

- ▶ tree  $\rightarrow$  the class
- ▶ forest  $\rightarrow$  the class, as resulting from a voting

# Nature of the prediction

Consider classification:

- ▶ tree → the class
  - ▶ “virginica” is just “virginica”
- ▶ forest → the class, as resulting from a voting
  - ▶ “241 virginica, 170 versicolor, 89 setosa” is different than “478 virginica, 10 versicolor, 2 setosa”

Different **confidence** in the prediction

## Bonus 3: confidence/tunability

Voting outcome:

- ▶ in classification, a measure of confidence of the decision
- ▶ in binary classification, voting threshold can be tuned to adjust bias towards one class (*sensitivity*)

**Q:** in regression?



## Subsection 3

### Binary classification

# Binary classification

Binary classification:

- ▶ one of the most common classes of problems
- ▶ (comparative) evaluation is important!

# Binary classification: evaluation

Consider the problem of classifying a person ('s data) as suffering or not suffering from a disease X.

Suppose we have “an accuracy of 99.99%”. **Q:** is it good?

# Binary classification: positives/negatives

Consider the problem of classifying a person ('s data) as suffering or not suffering from a disease X.

- ▶ **positive**: an observation of “suffering” class
- ▶ **negative**: an observation of “not suffering” class

In other problems, positive may mean a different thing: define it!

## Effectiveness indexes: FPR, FNR

Given some labeled data and a classifier for the disease X problem, we can measure:

- ▶ the number of negative observations *wrongly* classified as positives: False Positives (**FP**)
- ▶ the number of positive observations *wrongly* classified as negatives: False Negatives (**FN**)

To decouple FP, FN from data size:

$$\text{FPR} = \frac{\text{FP}}{N} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$
$$\text{FNR} = \frac{\text{FN}}{P} = \frac{\text{FN}}{\text{FN} + \text{TP}}$$

# Accuracy and error rate

Relation of FPR, FNR with accuracy and error rate

$$\text{Accuracy} = 1 - \text{Error Rate}$$

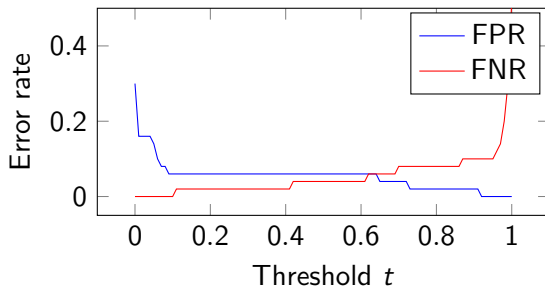
$$\text{Error Rate} = \frac{\text{FN} + \text{FP}}{\text{P} + \text{N}}$$

**Q:**  $\text{Error Rate} \stackrel{?}{=} \frac{\text{FPR} + \text{FNR}}{2}$

# FPR, FNR and sensitivity

- ▶ Suppose  $FPR = 0.06$ ,  $FNR = 0.04$  with threshold set to 0.5 (default for RF)
- ▶ One could be interested in “limiting” the FNR  $\rightarrow$  change the threshold

Experimentally:



# Comparing classifiers with FPR, FNR

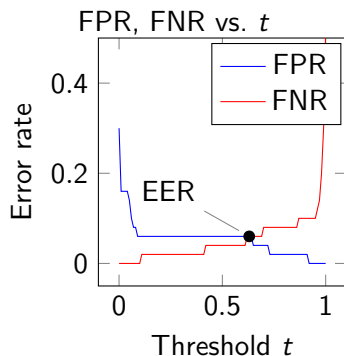
- ▶ Classifier A:  $\text{FPR} = 0.06$ ,  $\text{FNR} = 0.04$
- ▶ Classifier B:  $\text{FPR} = 0.10$ ,  $\text{FNR} = 0.01$

Which one is the better?

We'd like to have one single index  $\rightarrow$  EER, AUC

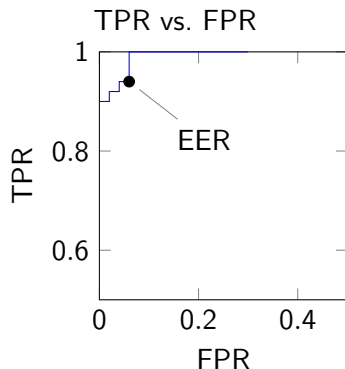


# Equal Error Rate (EER)



EER: the FPR at the value of  $t$  for which  $\text{FPR} = \text{FNR}$

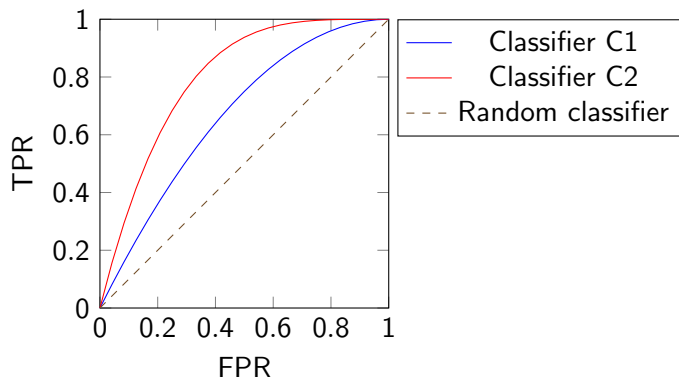
# AUC: Area Under the Curve



AUC: the area under the TPR vs. FPR curve, plotted for different values of threshold  $t$

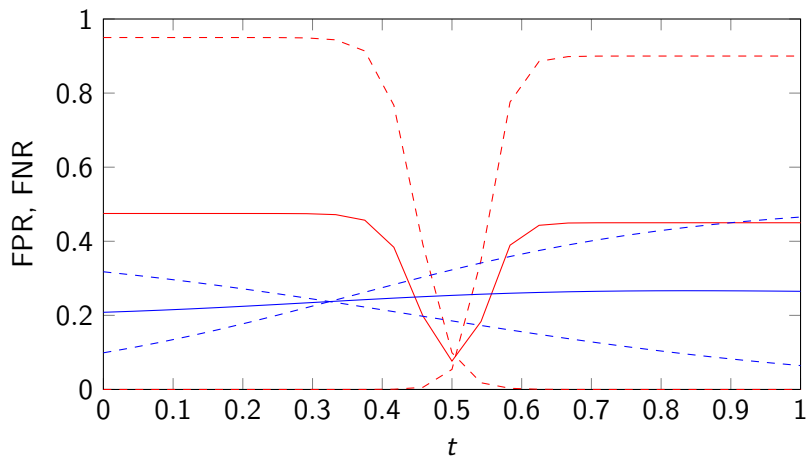
- ▶ the curve is called the *Receiver operating characteristic* (ROC)

# ROC and comparison



**Q:** what does the bisector represent?

## Other issues: robustness w.r.t. the threshold



“Same” with other parameters

## Other issues: robustness w.r.t. random components

Consider A vs. B, AUC measured with cross-fold validation:

- ▶ A: 0.85, 0.73, 0.91,  $\dots \rightarrow \mu = 0.83, \sigma = 0.15$
- ▶ B: 0.81, 0.78, 0.79,  $\dots \rightarrow \mu = 0.81, \sigma = 0.03$

Can we say that A is better than B? (for effectiveness only)

In general, other sources of performance variability:

- ▶ random seed
- ▶ subclass of problem class (e.g., image recognition of dogs, cats,  $\dots$ )

# Comparing techniques

Technique A, B; different index (e.g., AUC) values:

- ▶  $A \rightarrow (x_a^1, x_a^2, \dots) \rightarrow$  random variable  $X_a$
- ▶  $B \rightarrow (x_b^1, x_b^2, \dots) \rightarrow$  random variable  $X_b$

Do  $X_a, X_b$  follow different distributions?

- ▶ yes: A and B are different (concerning the AUC)
- ▶ no: difference in  $\mu_a, \mu_b$  might be due to randomness  $\rightarrow$  A, B are not *significantly* different

# Statistical significance in a nutshell

Just the way of thinking:

1. State a set of assumptions (the *null hypothesis*  $H_0$ ), e.g.:
  - ▶  $X_a, X_b$  are normally distributed and independent
  - ▶  $\bar{x}_a = \bar{x}_b$  (or  $\bar{x}_a \geq \bar{x}_b$ )
  - ▶ any other assumption in the *statistical model*

# Statistical significance in a nutshell

Just the way of thinking:

1. State a set of assumptions (the *null hypothesis*  $H_0$ ), e.g.:
  - ▶  $X_a, X_b$  are normally distributed and independent
  - ▶  $\bar{x}_a = \bar{x}_b$  (or  $\bar{x}_a \geq \bar{x}_b$ )
  - ▶ any other assumption in the *statistical model*
2. Perform a statistical test, appropriate choice depending on many factors, e.g.:
  - ▶ Wilcoxon test (many versions)
  - ▶ Friedman (many versions)
  - ▶ ...



# Statistical significance in a nutshell

Just the way of thinking:

1. State a set of assumptions (the *null hypothesis*  $H_0$ ), e.g.:
  - ▶  $X_a, X_b$  are normally distributed and independent
  - ▶  $\bar{x}_a = \bar{x}_b$  (or  $\bar{x}_a \geq \bar{x}_b$ )
  - ▶ any other assumption in the *statistical model*
2. Perform a statistical test, appropriate choice depending on many factors, e.g.:
  - ▶ Wilcoxon test (many versions)
  - ▶ Friedman (many versions)
  - ▶ ...
3. ... which outputs a  $p$ -value  $\in [0, 1]$ 
  - ▶ 0 is “good”, 1 is “bad”

## $p$ -value: meaning

0 is “good”, 1 is “bad”

The  $p$ -value is the degree to which the data conform to the pattern predicted by the null hypothesis

$$\blacktriangleright p\text{-value} = P(x_a^1, x_a^2, \dots, x_b^1, x_b^2, \dots | H_0)$$

If  $p$ -value is low:

- $\blacktriangleright$  we've been very (un)lucky in having observed  $x_a^1, x_a^2, \dots, x_b^1, x_b^2, \dots$
- $\blacktriangleright$  “maybe” because  $H_0$  is not true

## $p$ -value: meaning

0 is “good”, 1 is “bad”

The  $p$ -value is the degree to which the data conform to the pattern predicted by the null hypothesis

$$\blacktriangleright p\text{-value} = P(x_a^1, x_a^2, \dots, x_b^1, x_b^2, \dots | H_0)$$

If  $p$ -value is low:

- $\blacktriangleright$  we've been very (un)lucky in having observed  $x_a^1, x_a^2, \dots, x_b^1, x_b^2, \dots$
- $\blacktriangleright$  “maybe” because  $H_0$  is not true
  - $\blacktriangleright$  **Warning!** Any part of  $H_0$ , not necessarily the  $\bar{x}_a = \bar{x}_b$  part!

# Statistical significance

Things are much more complex than this. . .

Some interesting papers:

- ▶ Joaquín Derrac et al. “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18
- ▶ Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. “How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments”. In: *arXiv preprint arXiv:1806.08295* (2018)
- ▶ Sander Greenland et al. “Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations”. In: *European journal of epidemiology* 31.4 (2016), pp. 337–350

## Subsection 4

### Boosting

# Many views and aggregation

In bagging/RF (regression):

- ▶ many views are different samples
- ▶ aggregation is average

Alternative:

- ▶ many views are subsequent residuals
- ▶ aggregation is the sum

# Boosting

When learning:

1. Current data is learning data
2. Repeat  $B$  times
  - 2.1 learn a tree on current data
  - 2.2 current data becomes residuals of learned tree ( $\mathbf{y} - \hat{\mathbf{y}}$ )

When predicting:

1. Repeat  $B$  times
  - 1.1 get a prediction from  $i$ th learned tree
2. sum prediction

**Q:** implementation differences w.r.t. RF?

## Boosting (regression)

```
function BOOSTTREES( $\mathbf{X}, \mathbf{y}$ )  
   $t(\mathbf{X}) \leftarrow \mathbf{0}$   
  for  $i \in \{1, 2, \dots, B\}$  do  
     $t_i \leftarrow \text{BUILDDECISIONTREE}(\mathbf{X}, \mathbf{y}, d)$   
     $t(\mathbf{X}) \leftarrow t(\mathbf{X}) + \lambda t_i(\mathbf{X})$   
     $\mathbf{y} \leftarrow \mathbf{y} - \lambda t_i(\mathbf{X})$   
  end for  
  return  $t$   
end function
```

- ▶ Each learned tree should be simple (maximum splits  $d$ )
- ▶  $\lambda$  slows down learning

Trickier with classification.



# Boosting parameters

- ▶  $\lambda$  usually set to 0.01 or 0.001
- ▶  $\lambda$  and  $B$  interact: for small  $\lambda$ ,  $B$  should be large
- ▶ large  $B$  can lead to overfitting (unlike bagging/RF, **Q:** why)

Find a good value for  $B$  with cross-validation

(Both boosting and bagging general techniques)

# Bagging/RF/boosting in summary

	Tree	Bagging	RF	Boosting
interpretability	▲			
numeric/categorical	▲	▲	▲	▲
accuracy	▼		▲	▲
test error estimate		▲	▲	
variable importance		▲	▲	▲
confidence/tunability		▲	▲	
fast to learn	▲*			▼
(almost) non-parametric		▲	▲	

\*: **Q:** how faster? when? does it matter?

# Lab: visualize forest errors and boundaries (3 h)

Consider just *versicolor* and *virginica* and their classification:

1. investigate variable importance with RF
  - 1.1 verify that it is true by removing important variables
2. investigate influence of  $B$  (`ntree`)
3. compare against decision tree
4. plot fuzzy decision boundaries

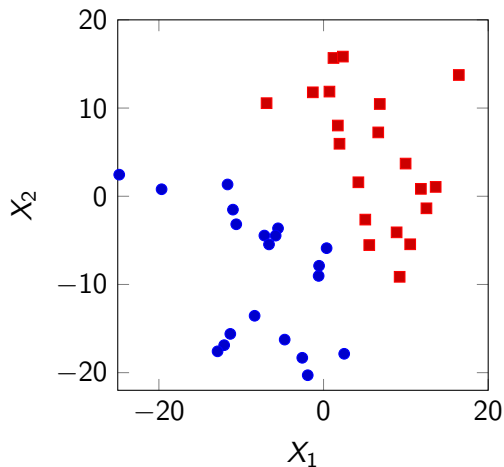
Packages: `randomForest`

Functions: `geom_raster`

## Section 6

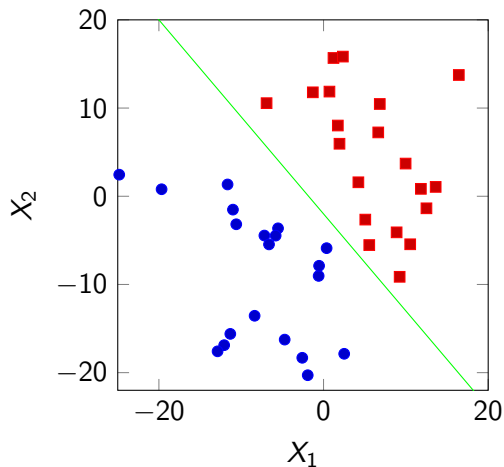
# Support Vector Machines

# Binary classification



Let's draw a decision boundary!

# Binary classification



Let's draw a decision boundary!

# Hyperplane

- ▶ We drew a line:

$$X_2 = mX_1 + q$$

- ▶ which can be written also as:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

- ▶ or, when the feature space is  $p$ -dimensional:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

the line is a *separating hyperplane*.

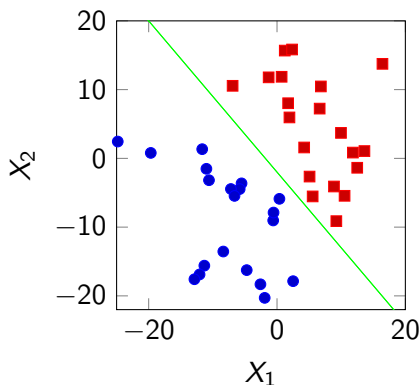
# Classification with a separating hyperplane

The hyperplane:

$$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$$

$$1.1X_1 + X_2 + 27 = 0$$

Given an observation  $(x_1, x_2)$ :





# Classification with a separating hyperplane

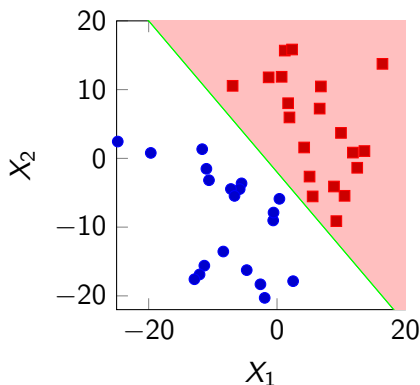
The hyperplane:

$$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$$

$$1.1X_1 + X_2 + 27 = 0$$

Given an observation  $(x_1, x_2)$ :

► if  $1.1X_1 + X_2 + 27 > 0$  then



# Classification with a separating hyperplane

The hyperplane:

$$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$$

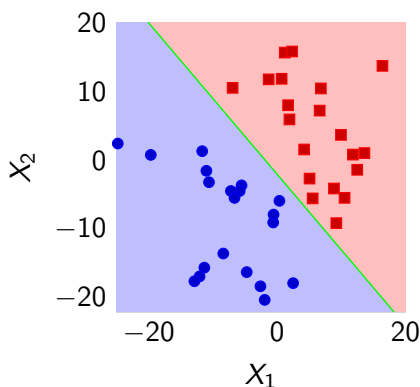
$$1.1X_1 + X_2 + 27 = 0$$

Given an observation  $(x_1, x_2)$ :

► if  $1.1X_1 + X_2 + 27 > 0$  then



► if  $1.1X_1 + X_2 + 27 < 0$  then



# Classification with a separating hyperplane

The hyperplane:

$$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$$

$$1.1X_1 + X_2 + 27 = 0$$

Given an observation  $(x_1, x_2)$ :

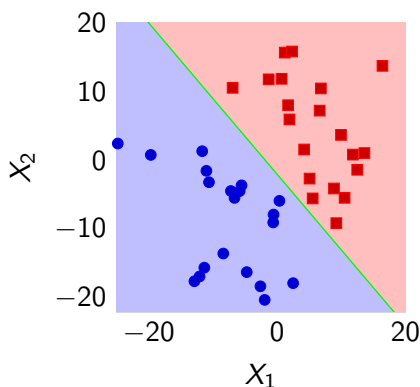
► if  $1.1X_1 + X_2 + 27 > 0$  then



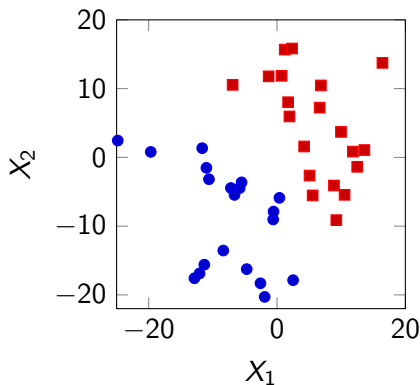
► if  $1.1X_1 + X_2 + 27 < 0$  then



The larger the difference, the stronger the confidence

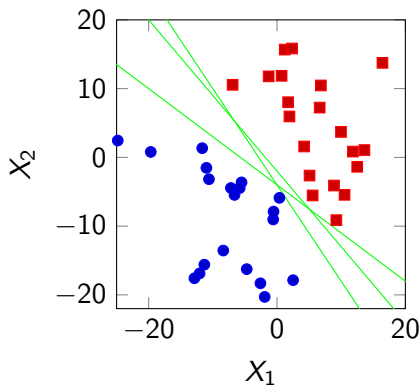


# Learning a separating hyperplane



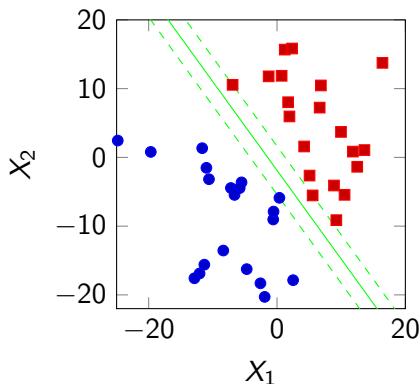
- ▶ We want an hyperplane which perfectly separates the learning data...

# Learning a separating hyperplane



- ▶ We want an hyperplane which perfectly separates the learning data...
- ▶ ...but there could be many ( $\infty$ ) of them! Which one?

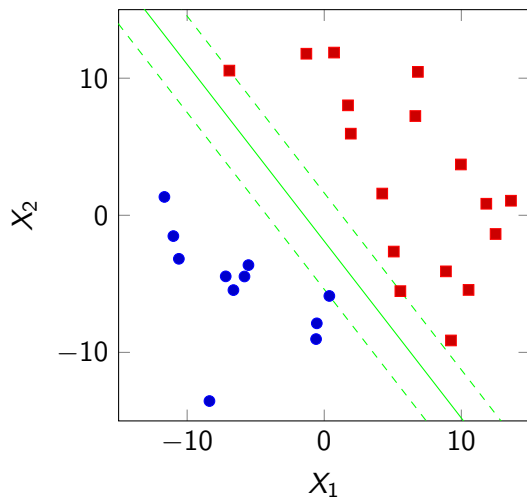
# Learning a separating hyperplane



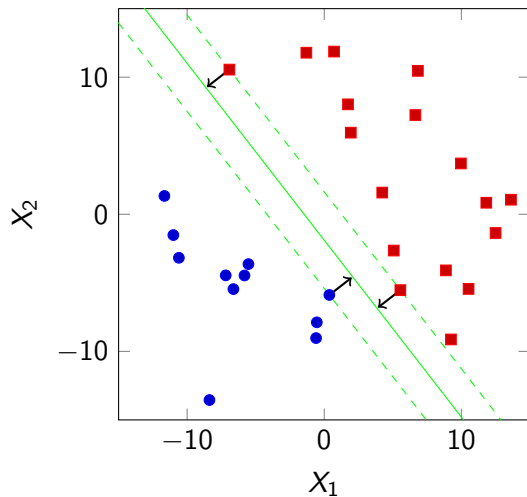
- ▶ We want an hyperplane which perfectly separates the learning data...
- ▶ ...but there could be many ( $\infty$ ) of them! Which one?
- ▶ Idea: the farthest from the learning observations!

**Maximal margin classifier**

# Maximal margin classifier



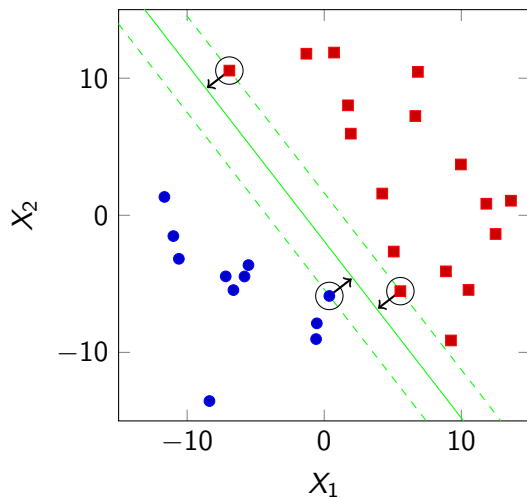
# Maximal margin classifier



► margin  $M$



# Maximal margin classifier



- ▶ margin  $M$
- ▶ *support vectors*

# Learning the maximal margin classifier

- ▶ Find the line which:
  1. perfectly separates learning observations
  2. has the largest margin from support vectors

Looks like an optimization problem...

# Learning the maximal margin classifier

$$\max_{\beta_0, \dots, \beta_p} M$$

under constraints

$$\sum_{j=1}^p \beta_j^2 = 1$$

$$\forall i \in \{1, \dots, n\}, y_i(\beta_0 + \beta_1 x_{i,1} + \dots + \beta_p x_{i,p}) \geq M$$

Some math tricks:

- ▶ if  $\sum_{j=1}^p \beta_j^2 = 1$ , then  $|\beta_0 + \beta_1 x_{i,1} + \dots + \beta_p x_{i,p}|$  is the distance between  $x_i^T$  and the hyperplane
- ▶ if  $y \in \{1, -1\}$ , then writing  $y_i(\dots) \geq M$  is like writing  $\dots \geq M, \forall \blacksquare$  and  $\dots \leq M, \forall \bullet$

## Support vectors

$$\forall i \in \{1, \dots, n\}, y_i(\beta_0 + \beta_1 x_{i,1} + \dots + \beta_p x_{i,p}) = M$$

They lie exactly on the margin!

# Learning the maximal margin classifier

Looks like an optimization problem...

...which is not hard to be solved.

# Maximal margin classifier issues

- ▶ What if the learning data is not perfectly separable?
  - ▶ cannot learn!
- ▶ What if a learning observation (being a support vector) is added/removed?
  - ▶ could learn a very different classifier → high variance!

# High variance of Maximal margin

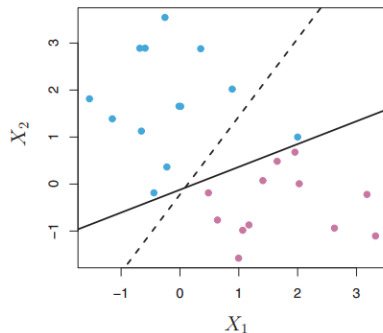
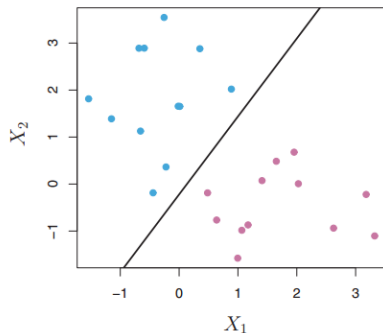


Image from An Introduction to Statistical Learning

# Soft margin

How to cope with these issues?

- ▶ Idea: be more tolerant!
  - ▶ some learning observation may be within the margin
  - ▶ some learning observation may be misclassified

Margin can be exceeded → *soft margin classifier* or *support vector classifier*



# Learning with toleration: support vector classifier

$$\max_{\beta_0, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n} M$$

under constraints

$$\sum_{j=1}^p \beta_j^2 = 1$$

$$\forall i \in \{1, \dots, n\}, y_i(\beta_0 + \beta_1 x_{i,1} + \dots + \beta_p x_{i,p}) \geq M(1 - \epsilon_i)$$

$$\forall i \in \{1, \dots, n\}, \epsilon_i \geq 0$$

$$\sum_{j=1}^n \epsilon_j = C$$

- ▶  $\epsilon_i$  are positive slack variables
  - ▶ if  $\epsilon_i \in ]0, 1[$ , then  $x_i^T$  is within the margin
  - ▶ if  $\epsilon_i \in [1, \infty[$ , then  $x_i^T$  is misclassified
- ▶  $C$  is the toleration budget ( $C = 0 \rightarrow$  maximal margin classifier)

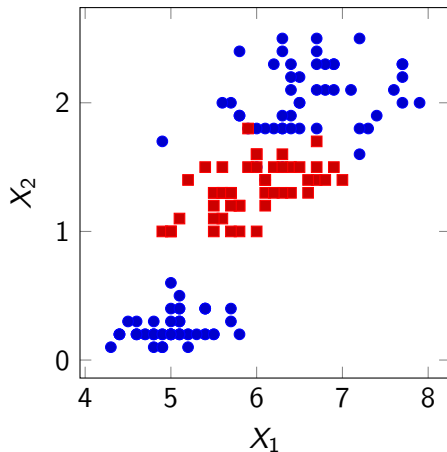
# Role of the parameter $C$

The larger  $C$

- ▶ the larger the toleration
- ▶ the larger the number of learning observations which can exceed the margin (or be misclassified)
- ▶ the larger the number of support vectors
- ▶ the lower the variance



# Linearity?



Some problems cannot be solved with an hyperplane!

## Some math rewriting

Finding values for  $\beta_0, \dots, \beta_p$  involves computing inner products between pair of observations:

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{i,j} x_{i',j}$$

And we can rewrite:

$$\beta_0 + \sum_{i=1}^p \beta_i x_i^* = f(x^*) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x^*, x_i \rangle$$

For non support vectors,  $\alpha_i = 0 \Rightarrow x_i$  does not impact on  $f(x^*)$ !

# Non support vectors

$$f(x^*) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x^*, x_i \rangle$$

- ▶  $f(x^*)$  is the distance of  $x^*$  from the decision boundary
- ▶ the (position of the) decision boundary depends only on the support vectors
- ▶  $\Rightarrow f(x^*)$  depends only on the support vectors

When predicting:

$$f(x^*) = \beta_0 + \sum_{i \in S} \alpha_i \langle x^*, x_i \rangle$$

# Kernel

Equation for the decision boundary can be generalized

$$f(x^*) = \beta_0 + \sum_{i=1}^n \alpha_i K(x^*, x_i)$$

Where  $K(x^*, x_i)$  is a function  $K : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ , called *kernel* (with some other properties).

## Support Vector Machines

# Intuition for the kernel

Consider prediction:

$$f(x^*) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x^*, x_i)$$

- ▶  $x^*$  is mapped from  $\mathbb{R}^p$  to  $\mathbb{R}^{p'}$ , with  $p' \gg p$ , using a function  $\Phi$ :  $K(x_i, x_j)$  computes the inner product  $\langle \phi(x_i), \phi(x_j) \rangle$  of mapped  $x_i, x_j$  without explicitly mapping them (*kernel trick*)
- ▶ the  $\alpha_i$  define (indirectly) an hyperplane in  $\mathbb{R}^{p'}$
- ▶ the classification is done by means of a separating hyperplane in the new space, i.e.,  $f(x^*)$  measures the distance of *mapped*  $x^*$  from the hyperplane



# Kernels

- ▶ linear kernel:

$$K(x^*, x_i) = \langle x_i, x^* \rangle = \sum_{j=1}^p x_{i,j} x_j^*$$

- ▶ polynomial kernel: ( $d$  is the degree)

$$K(x^*, x_i) = \left( 1 + \sum_{j=1}^p x_{i,j} x_j^* \right)^d$$

- ▶ radial basis function kernel (or radial, or RBF, or Gaussian):

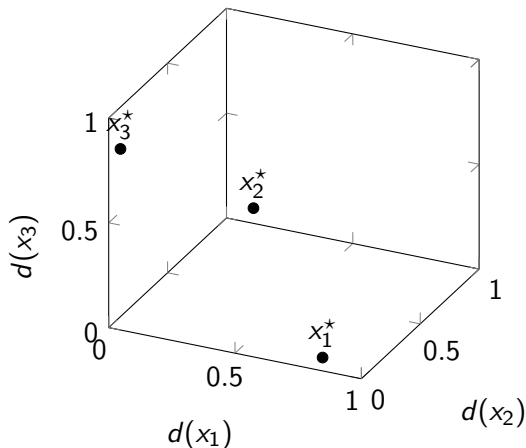
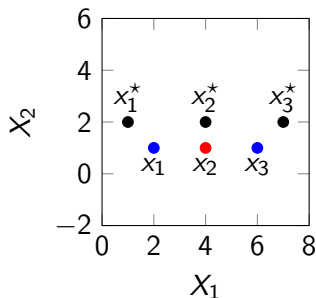
$$K(x^*, x_i) = \exp \left( -\gamma \sum_{j=1}^p (x_{i,j} - x_j^*)^2 \right)$$

## Intuition behind radial kernel

$$K(x^*, x_i) = \exp \left( -\gamma \sum_{j=1}^p (x_{i,j} - x_j^*)^2 \right) = \exp (-\gamma \|x_i, x^*\|^2)$$

- ▶ the coordinates in the new space are related to the distances of  $x^*$  from the support vectors (the closer, the higher the  $K(\cdot)$ ,  $K(\cdot) \in ]0, 1]$ )
- ▶  $\gamma$  determines how fast the coordinate goes to 0, i.e., a support vector becomes irrelevant for classifying  $x^*$

# Very raw visual intuition



$d(x_1)$  “means”  $\exp(-\gamma ||x_1, \cdot||^2)$

**Q:** draw a reasonable decision boundary, in the two spaces

# Multiclass ( $> 2$ ) classification with SVM

- ▶ one-vs.-one classification
- ▶ one-vs.-all classification

and many other proposals...

# One-vs.-one SVM

When learning:

1. for each pair  $(\mathcal{C}_1, \mathcal{C}_2)$  of classes, learn a binary SVM

When predicting:

1. for each learned SVM, predict class  $\hat{y}$
2. choose the most frequently predicted class

$$\binom{K}{2} = \frac{K(K-1)}{2} \text{ binary classifiers}$$

# One-vs.-all SVM

When learning:

1. for each class  $\mathcal{C}_i$ , learn a binary SVM ( $\mathcal{C}_i$  vs. all  $\mathcal{C}_j$ , with  $j \neq i$ ,  $\mathcal{C}_i$  coded as  $y = +1$ )

When predicting:

1. for each learned SVM, get  $f(x^*)$
2. choose the class with the largest  $f(x^*)$

$K$  classifiers

# Lab: SVM kernels (1 h)

Consider versicolor vs. virginica+setosa using only petal info:

1. use different kernels
2. play with params
3. artificially unbalance data and compare against RF

Packages: `e1071`

Functions: `plot` with SVM values

## Section 7

### Text mining



# Text mining

## Definition

**Text mining** is about extracting high level information from textual data.

A joint effort of *Machine Learning* and *Natural Language Processing* (NLP).

## Example 1: sentiment on brands

Is people on Twitter talking good or bad about brand *X*?

## Example 2: topics in letters

In this corpus of letters to/from the front in WW1, which are the topics covered?

# Common tasks

- ▶ text categorization
- ▶ text clustering
- ▶ entity extraction
- ▶ sentiment analysis
- ▶ summarization
- ▶ ...

## Example 3: relevance of citations

Find a way to quantify relevance of a citation from a scientific paper  $A$  to a scientific paper  $B$ ?

# Step 0

- ▶ Define the nature of the solution:
  - ▶ input
  - ▶ output
  - ▶ learning data (if any)
- ▶ Define a way to asses a solution

## Step 0: is it easy?

- ▶ **Q:** for “sentiment on brands”?
- ▶ **Q:** for “topics in letters”?
- ▶ **Q:** for “relevance of citations”?

# Natural language and ambiguity

- ▶ Text is (usually) natural language
- ▶ Natural means “as humans naturally express”  $\implies$  ambiguity!

Expect “raw results” to be worse than in normal ML!



## Subsection 1

### Sentiment analysis (and text categorization)

# Problem formalization

- ▶ Input: a piece of text (*document*)
- ▶ Output?
  - ▶ a numeric value in  $[-1, 1]$  (positivity)
  - ▶ a categorical value in  $\{\text{Pos}, \text{Neg}\}$
  - ▶ a categorical value in  $\{\text{Pos}, \text{Neutral}, \text{Neg}\}$
- ▶ **Q:** learning data?

Regression, multiclass classification, or binary classification (possibly with confidence).

We can use the techniques we already know (e.g., RF)!

## Which are the features?

Good coffee. Great for families. Always had good service. We go early so pretty empty. Flexible with menus. Wish they would remove service charge.

Need to transform a document into an numerical vector!

# Text to features

- ▶ Not only for sentiment analysis
- ▶ Many options
- ▶ Options can be combined

# Bag of words

- ▶ One dimension (feature, dependent variable) for each word
- ▶ Value of  $x_{i,j}$  is the number of occurrences of  $j$ -th word in the  $i$ -th document.

$t_1 = \text{the cat is on the table}$

$$x_{1,\text{the}} = 2$$

$$x_{1,\text{cat}} = 1$$

$$x_{1,\text{is}} = 1$$

$$x_{1,\text{mouse}} = 0$$

...

# Bag of which words?

- ▶ One dimension (feature, dependent variable) for each word

Which words? How big is  $x_i$ ?  $p = ?$

- ▶ common solution: the most  $k$  frequent words in the corpus

“Interesting” words not frequent enough in the corpus may be lost

# Stop words

- ▶ Some words may be very frequent, but useless for specific task (e.g., sentiment analysis)
  - ▶ a, an, the, are, ... (**stop words**)
- ▶ Just remove them!

Stop words are language dependent!

# Stemming

- ▶ There are variants for many words:
  - ▶ drink, drinks, drinking
  - ▶ happy, happier
- ▶ Even more in other languages:
  - ▶ mangio, mangia, mangi, mangiai, ...
- ▶ **Stemming**: reduce word to its word stem (the morphological root)
  - ▶ drinking → drink
  - ▶ argued → argu

Stemming are language dependent!



# A typical workflow

- ▶ Preprocessing ( $d \rightarrow d'$ )
  1. remove punctuation
  2. to lowercase
  3. remove stop words
  4. stemming
- ▶ Learning
  1. preprocess each  $d$  in corpus
  2. find most frequent  $k$  words in preprocessed corpus
  3. compute **X**
  4. learn a classifier
- ▶ Predicting
  1. preprocess input  $d$
  2. predict based on preprocessed  $d$

# Limitations and caveat: punctuation

- ▶ remove punctuation

It has been show that often punctuation matter (e.g., Twitter sentiment analysis):

- ▶ I just saw Alice.
- ▶ I just saw Alice!!!!
- ▶ I just saw Alice!!! :-)))))

# Case

- ▶ to lowercase

Case may be relevant in some case (e.g., music genre preferences classification):

- ▶ I like the Take That and I hate The Who.
- ▶ Who likes to take that song of Hate? Me!

# Goal, context, hypothesis

Twitter profiling: predict age and gender of user from his/her tweets. (Q: what kind of problem/problems?)

- ▶ people of different ages differently use case
- ▶ people of different age/gender differently use punctuation

A step in the workflow corresponds to an (implicit) hypothesis:

- ▶ remove stop words → stop words frequencies is not useful for predicting X

# Words that matter

Word count may be too coarse to capture desired information:

- ▶ documents with very different lengths
- ▶ irrelevant terms with general high frequencies

Use frequency or more complex variants

$$x_{i,j} = x_{d,t} = \text{tf}(t, d)\text{idf}(t, D)$$

- ▶  $\text{tf}(t, d) = f_{t,d}$ , term frequency
  - ▶ the more important the term  $t$  in document  $d$ , the larger
- ▶  $\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D: f_{t,d} > 0\}|}$ , inverse document frequency
  - ▶ the more common  $t$  in the corpus, the lower

# Bag of words and ordering

Sentiment analysis of restaurant reviews:

$t_1$  = The beer was good and the pub was not too noisy.

$t_2$  = The beer was not good and the pub was too noisy.

$x_1 = x_2$

- ▶ fundamental problem: ordering is lost
- ▶ even more fundamental: natural language can be hard to algorithmically understand (irony, sarcasm, ...)

Solutions:

- ▶ ngrams
- ▶ text parsing (NLP)

## Aside: collecting data for text classification

### Example: irony detection

- ▶ Pavel Savov and Radoslaw Nielek. “Ridiculously Expensive Watches and Surprisingly Many Reviewers: A Study of Irony”. In: *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*. IEEE. 2016, pp. 725–729



# ngrams

Instead of counting words, count short (up to  $n$ ) sequences of words:

- ▶  $x_{1,\text{dog,eat,cat}}$  instead of  $x_{1,\text{dog}}$
- ▶ size of data ( $p$ ) grows dramatically (and is sparser)
- ▶ useful in general for manipulating sequences

# Generality of a sentiment classifier

How many sentiment classifier should exist? Words that matter in sentiment should be predefined

- ▶ predefined list of opinion words (positive, negative), i.e., features are those words
- ▶ but context often matter
  - ▶ predictable is good for a car and bad for a movie
  - ▶ features for sentiment analysis in Twitter are likely different than features from sentiment analysis of a early '900 writer's correspondence

There are many pre-trained tool, often with more complex outcome than positive/negative.

# Out-of-the-box sentiment analysis

Should I use a pre-trained tool or build my own?

It depends:

- ▶ is sentiment analysis just a piece of a more complex ML system?
- ▶ which is my budget?
- ▶ is learning data easily available?

# Parsing: POS tagging

Assign a role to part of speech (**POS**):

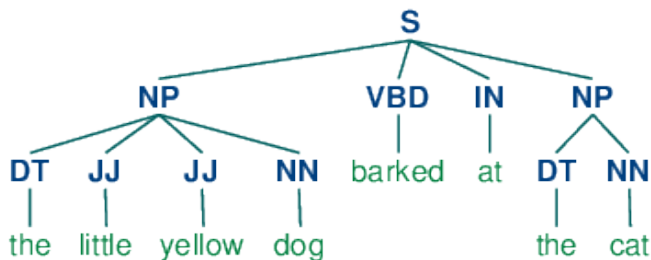


Image from <http://www.nltk.org/>

# Lab: Text categorization: sport vs. politics (4h)

Build a binary classifier for tweets: sport vs. politics

1. decide input, output
2. decide solution assessment
3. decide (if any) how to obtain learning data
4. decide workflow and ML technique

# R and Twitter

## Package `twitterR`

- ▶ (Linux, possibly install `libcurl4-gnutls-dev` and `libssl-dev`)
- ▶ needs Twitter API registration:  
<https://apps.twitter.com/>
- ▶ <https://rayli.net/blog/data/newborn-app-using-twitter-and-r-data-analysis/>
- ▶ `retrieve(userTimeline("MaleLabTs", n = 20))`, convert in DF  
`(twListToDF(tweets))`

# Text mining in R

## Package tm

- ▶ `http:`  
`//www.rdatamining.com/docs/text-mining-with-r`
- ▶ load data from character vector  
`Corpus (VectorSource(tweets.df$text))`
- ▶ to lowercase `tm_map(myCorpus, content_transformer(tolower))`
- ▶ remove punctuation  

```
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*",  
  , "", x)  
tm_map(myCorpus, content_transformer(removeNumPunct))
```
- ▶ remove stop words  
`tm_map(myCorpus, removeWords, myStopwords)`
- ▶ stemming `tm_map(myCorpus, stemDocument)` (requires package `snowballC`)
- ▶ getting as data  
`as.data.frame(t(as.matrix(TermDocumentMatrix(myCorpus))))`

## Subsection 2

Topic modeling (very very very briefly)



# Topics

Given a corpus of documents, what do they talk about?

- ▶ talks about  $\rightarrow$  *topic*

# Probabilistic model

Assume stochastic document building process:

- ▶ there exist  $k$  topics
- ▶ a topic is a distribution over words
- ▶ a topic is assigned to the document according to a known probability (a document may exhibit multiple topics)
- ▶ a word in a document is drawn according to topic and document-topic assignment

Words order does not matter!

# Probabilistic model

## Topics

gene 0.64  
dna 0.62  
genetic 0.61  
...

life 0.62  
evolve 0.61  
organism 0.61  
...

brain 0.64  
neuron 0.62  
nerve 0.61  
...

data 0.62  
number 0.62  
computer 0.61  
...

## Documents

### Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to survive? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using **computer** analyses to compare known **genomes**, concluded that today's **genomes** can be sustained with just 250 genes, and that the earliest life forms required a mere 128 **genes**. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those **predictions**

"are not all that far apart," especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson at Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic numbers** game, particularly as more and more **genomes** are completely sequenced. "It may be a way of organizing any newly **sequenced genome**," explains Arcady Mushegian, a **computational** molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

## Topic proportions and assignments

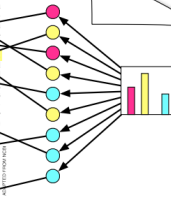


Image from <https://www.cs.princeton.edu/~blei/topicmodeling.html>

# Probabilistic graphical model

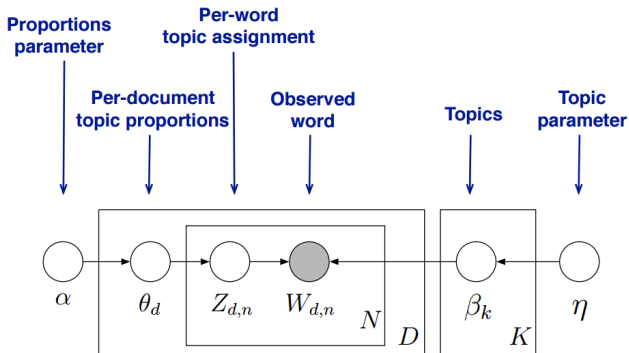


Image from <https://www.cs.princeton.edu/~blei/topicmodeling.html>

- ▶ nodes are random variables
- ▶ edges are dependencies
- ▶ shaded nodes are observed
- ▶ boxes are repeated variables

# Latent Dirichlet allocation (LDA)

A way for inferring distributions/assignments from observed values!  
(*Posterior inference*)

Given  $K$  (parameter),

- ▶ for each topic, compute words distribution
- ▶ for each document, compute topic “distribution”
- ▶ **Latent** refers to the unknown random variables
- ▶ **Dirichlet** is the distribution assumed for topics and words
- ▶ **Allocation** of words to topics and topics to documents

# LDA internals

(Just a coarse overview)

While inferring posterior, try to (both):

- ▶ associate each document with as few topics as possible
- ▶ associate each topic with as few words as possible

Conflicting goals, which results in finding (and putting in the same topics) words which often co-occur

# LDA output

- ▶ For each document of the corpus, a vector in  $[0, 1]^K$  where  $i$ -th value is “how much the document exhibits  $i$ -th topic”
  - ▶ reasonable values for the number of topics  $K$  is some tens (10–50) (**Q**: how to choose the right value for a problem?)
- ▶ For each topic, a vector  $[0, 1]^V$  where the  $i$ -th value is “how much the  $i$ -th word (on  $V$  words) is associated with the topic”
  - ▶ how to visualize/understand a topic? Select its most likely words

# Visualize topics



Image from <https://www.cs.princeton.edu/~blei/topicmodeling.html>



# LDA as a building block

- ▶ corpus visualization
- ▶ document similarities
- ▶ ...
- ▶ document  $\rightarrow \mathbb{R}^K$

LDA: document  $\rightarrow \mathbb{R}^K$

How to apply to new data?

- ▶ assume everything is known (i.e., already computed on the corpus)
- ▶ just infer the posterior of topic assignment for the new document

## Lab: Sport vs. politics with topics (1.5h)

Augment the classifier of previous lab with LDA

In R:

- ▶ package `topicmodels`
- ▶ functions `LDA(train.data, k)`, `posterior(lda.model, test.data)`

## Section 8

### Recommender systems

# Recommender systems

Scenario: a service where users consume items

- ▶ predict the users' rating to unconsumed items

# Recommender systems

Scenario: a service where users consume items

- ▶ predict the users' rating to unconsumed items
- ▶ great interest from industry
  - ▶ e-commerce
  - ▶ online social networks
  - ▶ entertainment on demand
- ▶ users usually pay for consuming

## Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses				
Sweetness day				
The true love				
Crazy Max				
The final judgement				

## Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses	5	5	0	0
Sweetness day	5			0
The true love		4	0	
Crazy Max	0	0	5	4
The final judgement	0	0	5	

- ▶ some users rated some movies



## Example: movie recommendation

Movie	Alice	Bob	Carol	Dave
Hugs and kisses	5	5	0	0
Sweetness day	5	?	?	0
The true love	?	4	0	?
Crazy Max	0	0	5	4
The final judgement	0	0	5	?

- ▶ some users rated some movies
- ▶ predict the rating of unrated movies
  - ▶ **Q:** and then? where's the recommendation?

# Content-based representation

Suppose 2 features (independent variables) exist for movies (*content of*)

- ▶  $X_1$  represents “romance”
- ▶  $X_2$  represents “action”
- ▶  $X_0 = 1$  represents bias

Movie	Alice	Bob	Carol	Dave	$X_1$	$X_2$
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

## Notation

Movie	Alice	Bob	Carol	Dave	$X_1$	$X_2$
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

- ▶  $r_{i,j} \in \{0, 1\}$  is 1 iff user  $j$  rated movie  $i$
- ▶  $y_{i,j}$  is rating given by user  $j$  to movie  $i$  (iff  $r_{i,j} = 1$ )
- ▶  $x_i$  is the feature vector of movie  $i$

## Notation

Movie	Alice	Bob	Carol	Dave	$X_1$	$X_2$
Hugs and kisses	5	5	0	0	0.95	0.01
Sweetness day	5	?	?	0	1	0
The true love	?	4	0	?	0.99	0
Crazy Max	0	0	5	4	0	1
The final judgement	0	0	5	?	0.02	0.99

- ▶  $r_{i,j} \in \{0, 1\}$  is 1 iff user  $j$  rated movie  $i$
- ▶  $y_{i,j}$  is rating given by user  $j$  to movie  $i$  (iff  $r_{i,j} = 1$ )
- ▶  $x_i$  is the feature vector of movie  $i$

“predict the rating of unrated movies” corresponds to **solving**  $n_u$  **(number of users) regression problems**

- ▶ learn  $f_{\text{Alice}}(x)$ ,  $f_{\text{Bob}}(x)$ ,  $\dots$

# Recommendation as linear regression

Assume a linear dependency between rating and features:

$$\begin{aligned}y_{i,j} &= \theta_{0,j}x_{i,0} + \theta_{1,j}x_{i,1} + \theta_{2,j}x_{i,2} + \dots \\ &= \theta_j^T x_i\end{aligned}$$

- ▶  $\theta_j \in \mathbb{R}^p$  is the set of parameters of user  $j$ 
  - ▶  $\theta_j$  represents preferences of user  $j$

“solving  $n_u$  (number of users) regression problems” corresponds to  
**for each user  $j$ , learn  $\theta_j$**

## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n \left( \theta_j^T x_i - y_{i,j} \right)^2$$

## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n \left( \theta_j^T x_i - y_{i,j} \right)^2$$

- ▶ minimize sum of squared errors

## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2$$

- ▶ minimize sum of squared errors
- ▶ consider only rated movies ( $r_{i,j} = 0$  for unrated)



## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^p \theta_{k,j}^2$$

- ▶ minimize sum of squared errors
- ▶ consider only rated movies ( $r_{i,j} = 0$  for unrated)
- ▶ with regularization (**Q**: role of  $\lambda$ ?)

## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \theta_j^T \theta_j$$

- ▶ minimize sum of squared errors
- ▶ consider only rated movies ( $r_{i,j} = 0$  for unrated)
- ▶ with regularization (**Q**: role of  $\lambda$ ?)

## Learning $\theta_j$

Goal: learned  $\theta_j$  should be minimize errors on already rated movies:

$$\min_{\theta_j} \frac{1}{2} \sum_{i=1}^n r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \theta_j^T \theta_j$$

- ▶ minimize sum of squared errors
- ▶ consider only rated movies ( $r_{i,j} = 0$  for unrated)
- ▶ with regularization (**Q**: role of  $\lambda$ ?)

For all users:

$$\min_{\theta_1, \dots, \theta_{n_u}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i=1}^n r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

Can be solved with any optimization algorithm, e.g., gradient descent

# The movie features?

- ▶ Which features to characterize movies? (or songs, products, people, ...)
- ▶ Who assign feature values to movies? Is it costly?
  - ▶ scale?

# The movie features?

- ▶ Which features to characterize movies? (or songs, products, people, ...)
- ▶ Who assign feature values to movies? Is it costly?
  - ▶ scale?

Assume we “want”  $p$  features:

Movie	Alice	Bob	Carol	Dave	$X_1$	...	$X_p$
Hugs and kisses	5	5	0	0	?	?	?
Sweetness day	5	?	?	0	?	?	?
The true love	?	4	0	?	?	?	?
Crazy Max	0	0	5	4	?	?	?
The final judgement	0	0	5	?	?	?	?

# Collaborative filtering

In content-based:

- ▶ we know movie features  $x_1, x_2, \dots, x_n$  and ratings  $y_1, y_2, \dots, y_{n_u}$
- ▶ we learn users' preferences  $\theta_1, \theta_2, \dots, \theta_{n_u}$

# Collaborative filtering

In content-based:

- ▶ we know movie features  $x_1, x_2, \dots, x_n$  and ratings  $y_1, y_2, \dots, y_{n_u}$
- ▶ we learn users' preferences  $\theta_1, \theta_2, \dots, \theta_{n_u}$

Assume we know users' preferences:

- ▶ we learn movie features

# Collaborative filtering

In content-based:

- ▶ we know movie features  $x_1, x_2, \dots, x_n$  and ratings  $y_1, y_2, \dots, y_{n_u}$
- ▶ we learn users' preferences  $\theta_1, \theta_2, \dots, \theta_{n_u}$

Assume we know users' preferences:

- ▶ we learn movie features

Users (implicitly) **collaborate** to characterize content



# Learning features from preferences

One movie:

$$\min_{x_i} \frac{1}{2} \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} x_i^T x_i$$

All movies:

$$\min_{x_1, \dots, x_n} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^n x_i^T x_i$$

# Users' preferences

“Assume we want  $p$  features” corresponds to “users' preferences are  $p$  dimensional”

- ▶ how to collect users' preferences?
- ▶ how many?
- ▶ relation with “linear dependency assumption”?

# Users' preferences

“Assume we want  $p$  features” corresponds to “users' preferences are  $p$  dimensional”

- ▶ how to collect users' preferences?
- ▶ how many?
- ▶ relation with “linear dependency assumption”?

It may be preferable to learn features and preferences together!

## Learning features and preferences

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_n, \theta_1, \dots, \theta_{n_u}} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T \mathbf{x}_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^n \mathbf{x}_i^T \mathbf{x}_i + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

## New user?

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$$\min_{\mathbf{x}_1, \dots, \mathbf{x}_n, \theta_1, \dots, \theta_{n_u}} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T \mathbf{x}_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^n \mathbf{x}_i^T \mathbf{x}_i + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

## New user?

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$$\min_{x_i, \dots, x_n, \theta_1, \dots, \theta_{n_u}} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^n x_i^T x_i + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

- ▶ sum of squared errors is always zero for the new user

## New user?

Movie	Alice	Bob	Carol	Dave	Eric
Hugs and kisses	5	5	0	0	?
Sweetness day	5	?	?	0	?
The true love	?	4	0	?	?
Crazy Max	0	0	5	4	?
The final judgement	0	0	5	?	?

$$\min_{x_i, \dots, x_n, \theta_1, \dots, \theta_{n_u}} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^{n_u} r_{i,j} \left( \theta_j^T x_i - y_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^n x_i^T x_i + \frac{\lambda}{2} \sum_{j=1}^{n_u} \theta_j^T \theta_j$$

- ▶ sum of squared errors is always zero for the new user
- ▶ the only goal is to minimize “sum of preferences”
  - ▶ **Q:** what happens without regularization?
- ▶ which results in no preferences ( $\forall_k, \theta_{k,j} = 0$ ), and hence equal predicted ratings for all movies

# Cold start problem

When something “new” arrives and no data is available:

- ▶ new user
- ▶ new movie (**Q:** is really a problem?)

One possible solution: use mean values

$$\mathbf{y}_{j'} = \frac{1}{n_u} \sum_{j=1}^{n_u} \mathbf{y}_j$$



# Recommender system assessment

Many options:

- ▶ as a regression problem, RMSE

# Recommender system assessment

Many options:

- ▶ as a regression problem, RMSE
- ▶ as a classification problem, accuracy
  - ▶ does the tool recommend the most preferred item to the user?

# Recommender system assessment

Many options:

- ▶ as a regression problem, RMSE
- ▶ as a classification problem, accuracy
  - ▶ does the tool recommend the most preferred item to the user?
- ▶ as a classification problem, accuracy@ $K$ 
  - ▶ does the tool recommend the most preferred item to the user among the top  $k$  recommendations?

# Recommender system assessment

Many options:

- ▶ as a regression problem, RMSE
- ▶ as a classification problem, accuracy
  - ▶ does the tool recommend the most preferred item to the user?
- ▶ as a classification problem,  $\text{accuracy}@K$ 
  - ▶ does the tool recommend the most preferred item to the user among the top  $k$  recommendations?
- ▶ as an information retrieval problem, *precision* and *recall*
  - ▶ does the tool recommend *relevant* items?

$$\text{Prec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{recomm.}} \quad \text{Rec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{relevant}}$$

**Q:** relation with FPR, FNR?

# Recommender system assessment

Many options:

- ▶ as a regression problem, RMSE
- ▶ as a classification problem, accuracy
  - ▶ does the tool recommend the most preferred item to the user?
- ▶ as a classification problem,  $\text{accuracy}@K$ 
  - ▶ does the tool recommend the most preferred item to the user among the top  $k$  recommendations?
- ▶ as an information retrieval problem, *precision* and *recall*
  - ▶ does the tool recommend *relevant* items?

$$\text{Prec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{recomm.}} \quad \text{Rec.} = \frac{\#(\text{relevant} \wedge \text{recomm.})}{\#\text{relevant}}$$

**Q:** relation with FPR, FNR?

In practice, how to measure them?

# Beyond accuracy

- ▶ diversity
- ▶ serendipity
  - ▶ positive surprise
- ▶ revenue? number of click/user/usages?

More in general, UI plays a crucial role!

## Section 9

### Evolutionary computation

# What is Machine Learning?

## Definition

**Machine Learning** is the science of getting computer to learn without being explicitly programmed.



## Up to now

“learn without being explicitly” → refine some predefined more general solution scheme

- ▶ RF for regression → find a *good* forest
- ▶ SVM for binary classification → find a *good* hyperplane

## Up to now

“learn without being explicitly” → refine some predefined more general solution scheme

- ▶ RF for regression → find a *good* forest
- ▶ SVM for binary classification → find a *good* hyperplane

We have some (quite precise) idea (the *hypothesis*) about the nature of the solution: a tree, an hyperplane, ...

What if we do not?

# No hypothesis

- ▶ We just have a way to assess a candidate solution
- ▶ No hypothesis
- ▶ Computer: be free, learn a (good) solution!

How?

# No hypothesis

- ▶ We just have a way to assess a candidate solution
- ▶ No hypothesis
- ▶ Computer: be free, learn a (good) solution! (= program yourself!)

How? A significant case:

- ▶ problem: life
- ▶ user: God?
- ▶ computer: nature
- ▶ learning method: natural evolution

# Evolutionary process

A general and basic scheme:

- ▶ a population of individuals compete for limited resources
- ▶ the population is dynamic: individuals die and are born
- ▶ fittest individual survive and reproduce more than the others
- ▶ offspring inherit some characters from parents (they are similar but not identical)

# Evolutionary process

A general and basic scheme:

- ▶ a population of individuals compete for limited resources
- ▶ the population is dynamic: individuals die and are born
- ▶ fittest individual survive and reproduce more than the others
- ▶ offspring inherit some characters from parents (they are similar but not identical)

On/by/for computers? Evolutionary computation (EC)

# EC: a bit of history

1930s first ideas

1960s ideas development using first computers

1970s exploration

1980s exploitation

1990s unification

2000s+ mature expansion

# Communities

At least three communities:

- ▶ biologists: simulate/understand real evolution
- ▶ computer scientists/engineers: build interesting artifacts
- ▶ artificial-life researchers: build/study artificial worlds

Result:

- ▶ some duplications
- ▶ different vocabularies
- ▶ strong habits

Kenneth A De Jong. *Evolutionary computation: a unified approach*. MIT press, 2006



# What can be taught/learned?

Here:

- ▶ general scheme
- ▶ terminology
- ▶ some significant variants
- ▶ general usage guidelines

Not here:

- ▶ (variant) details
- ▶ detailed motivation (“theory”)
- ▶ specific tools

# General scheme

- ▶ a population of individuals compete for limited resources
- ▶ the population is dynamic: individuals die and are born
- ▶ fittest individual survive and reproduce more than the others
- ▶ offspring inherit some characters from parents (they are similar but not identical)

Some questions:

- ▶ what is an individual?
- ▶ what is a population? what are resources?
- ▶ how individuals compete?
- ▶ how fitness is measured?
- ▶ how do individual reproduce?

# Individual

A candidate solution for the considered problem:

- ▶ a program in a given programming language
- ▶ a set of numerical parameters
- ▶ a picture
- ▶ ...

Internally represented as:

- ▶ itself (program, set, picture, ...)
- ▶ some well defined data structure:
  - ▶ a fixed/variable-length string of bits
  - ▶ an abstract syntax tree
  - ▶ ...

# Individual

A candidate solution for the considered problem: (**phenotype**)

- ▶ a program in a given programming language
- ▶ a set of numerical parameters
- ▶ a picture
- ▶ ...

Internally represented as: (**genotype**)

- ▶ itself (program, set, picture, ...)
- ▶ some well defined data structure:
  - ▶ a fixed/variable-length string of bits
  - ▶ an abstract syntax tree
  - ▶ ...

# Individual: why genotype/phenotype?

- ▶ To resemble nature
- ▶ To ease manipulation
  - ▶ how two programs should reproduce?
  - ▶ how two images should reproduce?
- ▶ To allow reuse, hence enabling actual usage of EC
  - ▶ someone found a good way of making bits strings reproduce
  - ▶ user “just” need to decide how to transform  
(**genotype-phenotype mapping**) a bits string to his/her  
solution form (e.g., numerical parameters)

# Population and competition for resources

Mainstream:

- ▶ a population is a set of individuals with a fixed (max) size
- ▶ “limited resources” is a place in the population

The population is dynamic:

- ▶ when a new individual is born, some individual must leave the population (die): which one?

# Population dynamics

How/when individuals are replaced? (**generational model** or replacement strategy)

Underlying (and common) assumptions:

- ▶ individuals life is instantaneous
  - ▶ given the genotype, the phenotype (if any) and the fitness are immediately known
- ▶ time flowing is determined by births (and deaths)

# Generational model: general scheme

Parameters:

- ▶ a population of  $m$  parents
- ▶ a population of  $n$  offspring (built from parents; how? later)
- ▶ a boolean flag (overlapping vs. non-overlapping)

(Recall: population size is fixed)



# Overlapping generational model

At each time tick:

1. build  $n$  offspring from the  $m$  parents
2. obtain an  $n + m$  population by merging parents and offspring
3. select  $m$  individuals to survive

# Non-overlapping generational model

At each time tick:

1. build  $n$  offspring from the  $m$  parents (assume  $n \geq m$ )
2. select  $m$  individuals to survive among the  $n$  offspring

All parents die!

## Common cases

- ▶  $n = m$ , overlapping
- ▶  $n = m$ , non-overlapping
- ▶  $n = 0.8m$ , overlapping
- ▶  $n = 1$ , overlapping (steady state)

## Common cases

- ▶  $n = m$ , overlapping
- ▶  $n = m$ , non-overlapping
- ▶  $n = 0.8m$ , overlapping
- ▶  $n = 1$ , overlapping (steady state)

Problem:

- ▶ different degrees of dynamicity in the single time tick
  - ▶ makes different variants comparison difficult

Solution:

- ▶ measure time flowing as number of births referred to population size  $m$
- ▶ a **generation** occurs each  $m$  births

# Selection criteria

How to

- ▶ select individuals to survive?
- ▶ select parents to reproduce?

Many options:

- ▶ uniform (neutral) selection
- ▶ fitness-proportional selection
- ▶ rank-proportional selection
- ▶ truncation selection
- ▶ tournament selection
- ▶ ...

# Fitness/rank-proportional

Fitness-proportional:

1. given the numerical fitness of each individual
2. randomly pick one individual with probability proportional to the fitness (the better, the larger probability)

Rank-proportional:

1. given the rank of each individual in a fitness-based ranking
2. randomly pick one individual with probability proportional to the rank (the better, the larger probability)

(Can be applied to a non-numerical fitness, in principle)

# Uniform and truncation

Uniform:

1. pick randomly an individual (with uniform probability)

Truncation:

1. pick the best individual (**elitism**)

(Deterministic)

# Tournament selection

Given a parameter  $n_{\text{size}}$  (size of the tournament):

1. randomly (with uniform probability) pick  $n_{\text{size}}$  individuals
2. from them, choose the one with the best fitness



# Selection criteria differences

Is criterion A better than criterion B? *Just* measure!

Criteria differ in how strongly they tend to prefer fit vs. unfit individuals:

- ▶ uniform selection: no preferences
- ▶ truncation selection: strong preference of fit individuals
- ▶ tournament:  $n_{\text{size}} \rightarrow 1$ : no preference,  $n_{\text{size}} \rightarrow m$ : strong preference

# Selecting fit/unfit individuals

Strong preference (or selective/evolutionary pressure):

- ▶ population tends to converge to fittest individuals
- ▶ evolution concentrates in improving most promising solutions (**exploitation**)
- ▶ risk of “falling” in local optimum

Weak preference (or selective/evolutionary pressure):

- ▶ population includes also unfit individuals
- ▶ evolution investigates many different (maybe not promising) solutions (**exploration**)
- ▶ risk of not finding a good solution

Exploration/exploitation trade-off is hard to rule!

## Selectors: common cases

- ▶ Reproduction: tournament of  $n_{\text{size}}$ 
  - ▶ e.g.,  $m = n_{\text{pop}} = 500$ ,  $n_{\text{size}} = 5$
- ▶ Survival: truncation
- ▶ Reproduction: fitness proportional
- ▶ Survival: truncation

# Reproduction

Build  $n$  offspring from the  $m$  parents. How?

General scheme:

- ▶ given one or more parents, an offspring is generated by applying a unary or binary **genetic operator** on parent genotypes
  - ▶ unary (**mutation**):  $f : \mathcal{G} \rightarrow \mathcal{G}$
  - ▶ binary (recombination or **crossover**):  $f : \mathcal{G}^2 \rightarrow \mathcal{G}$
- ▶ given  $n$  and a set of weighted operators, generate offspring with operators according to their weights (deterministically or stochastically)

# Choice of operators

Operators:

- ▶ crossover for generating 80% of offspring
- ▶ mutation for generating 20% of offspring

Deterministically:

1. for  $0.8n$  times
  - 1.1 select 2 parents (with reproduction selection criterion)
  - 1.2 apply crossover to genotypes
2. for  $0.2n$  times
  - 2.1 select 1 parent (with reproduction selection criterion)
  - 2.2 apply mutation to genotype

# Choice of operators

Operators:

- ▶ crossover for generating 80% of offspring
- ▶ mutation for generating 20% of offspring

Stochastically:

1. for  $n$  times
  - 1.1 randomly choose between mutation/crossover with 20/80 probability
  - 1.2 select 1 or 2 parents (with reproduction selection criterion) accordingly
  - 1.3 apply operator to genotype(s)

# Mutation for bits string genotypes

Most classical option: probabilistic bit flip mutation

1. copy parent genotype  $g_p$  as child genotype  $g_c$
2. for each bit in the in  $g_c$ , flip it ( $0 \rightarrow 1$  or  $1 \rightarrow 0$ ) with  $p$  probability

Commonly,  $p = 0.01$

$g_p = 0010100111101010101100100101$

$g_c = 0010101111101010101101100101$

# Crossover for (bits) string genotypes

Many options:

- ▶ one-point crossover
- ▶ two-points crossover
- ▶  $n$ -points crossover
- ▶ uniform crossover
- ▶ ...



# One-, two-, $n$ -points crossover

Assume parents with equal genotype size:

1. choose randomly one (two,  $n$ ) *cut points* in the genotype (indexes  $i$  such that  $i < |g_{p_1}| = |g_{p_2}|$ )
2. child bits before the cut point comes from parent 1, child bits after the cut point comes from parent 2

In general,  $j$ th bit comes from parent 1 iff closest larger cut point is even, from 2, otherwise.

# One-, two-, $n$ -point crossover

One-point:

$$g_{p_1} = 001010011110101010|1100100101$$

$$g_{p_2} = 11101010101001010|0101110111$$

$$g_c = 001010011110101010 \text{ } 0101110111$$

Two-points:

$$g_{p_1} = 0010100|1110101010|1100100101$$

$$g_{p_2} = 1110101|0101001010|0101110111$$

$$g_c = 0010100 \text{ } 0101001010 \text{ } 1100100101$$

# Uniform crossover

A cut point is placed at each index with  $p = 0.5$  probability

# Crossover with variable length (bits) string genotype

Many variants:

- ▶ one-, two-points crossover
  - ▶ cut points may be different within parents
  - ▶ child genotype size may be larger or smaller than parents sizes
- ▶ ...

One-point:

$$g_{p_1} = 00101001110101010|1100100101$$

$$g_{p_2} = 111010101|010010100101110111$$

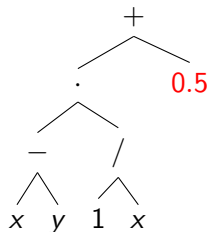
$$g_c = 00101001110101010 \ 010010100101110111$$

Genotype-phenotype mapping must allow for variable length genotypes!

# Mutation (trees)

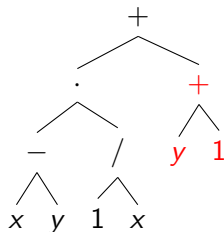
Parent

$$(x - y)\frac{1}{x} + 0.5$$



Child

$$(x - y)\frac{1}{x} + 1 + y$$



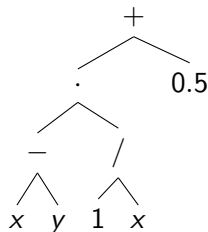
1. choose a random subtree
2. replace with a randomly generated subtree

Usually, constraints on depth

# Crossover (trees)

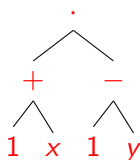
Parent 1

$$(x - y) \frac{1}{x} + 0.5$$



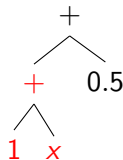
Parent 2

$$(1 + x)(1 - y)$$



Child

$$1 + x + 0.5$$



1. choose a random subtree in parent 1
2. choose a random subtree in parent 2
3. swap subtrees (child is copy of parent)

Usually, constraints on depth

# Role of operators

Mutation (x) or crossover?

- ▶ mutation  $\rightarrow$  exploitation
- ▶ crossover  $\rightarrow$  exploration

# Population initialization

- ▶ Totally random
- ▶ More specific approaches, dependent on genotype form



# Fitness

Fitness of an individual = ability to solve the problem of interest

- ▶ errors on several fitness cases by execution/simulation/application

Common cases:

- ▶ one numerical index
- ▶ more than one numerical indexes
- ▶ ...

Closely related with selectors

# Many indexes: multiobjective

$$f(i) = \langle f_1(i), \dots, f_n(i) \rangle$$

How to compare individuals  $i_1, i_2$  ?

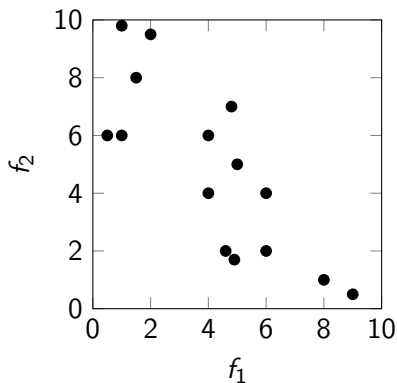
- ▶ linearization
  - ▶  $f(i) = \alpha_1 f_1(i) + \dots + \alpha_n f_n(i)$
- ▶ lexicographical order
  - ▶ compare  $f_1(i_1) \stackrel{?}{>} f_1(i_2)$ ; if tie,  $f_2(i_1) \stackrel{?}{>} f_2(i_2)$ ; ...
- ▶ Pareto dominance
- ▶ ...

**Q:** with which selectors?

# Pareto dominance

$i_1$  dominates  $i_2$  iff:

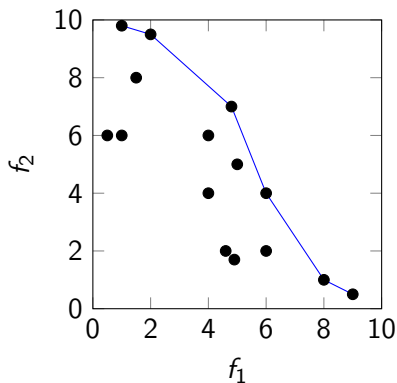
$$\forall j, f_j(i_1) \geq f_j(i_2) \wedge \exists k, f_k(i_1) > f_k(i_2)$$



# Pareto dominance

$i_1$  dominates  $i_2$  iff:

$$\forall j, f_j(i_1) \geq f_j(i_2) \wedge \exists k, f_k(i_1) > f_k(i_2)$$

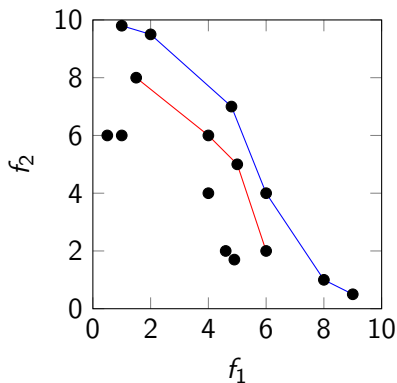


► 1st Pareto front:  
undominated solutions

# Pareto dominance

$i_1$  dominates  $i_2$  iff:

$$\forall j, f_j(i_1) \geq f_j(i_2) \wedge \exists k, f_k(i_1) > f_k(i_2)$$

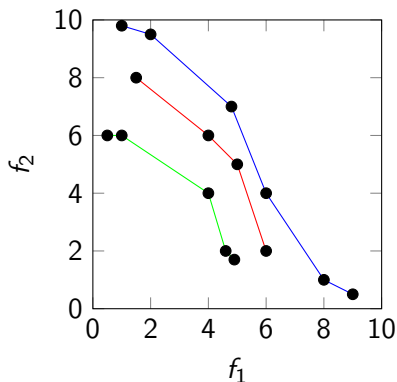


- ▶ 1st Pareto front:  
undominated solutions
- ▶ 2nd Pareto front:  
undominated solutions,  
while not considering 1st  
front

# Pareto dominance

$i_1$  dominates  $i_2$  iff:

$$\forall j, f_j(i_1) \geq f_j(i_2) \wedge \exists k, f_k(i_1) > f_k(i_2)$$



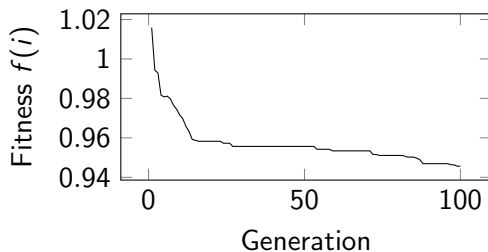
- ▶ 1st Pareto front:  
undominated solutions
- ▶ 2nd Pareto front:  
undominated solutions,  
while not considering 1st  
front
- ▶ ...

## In practice

- ▶ Is my EA working?
- ▶ When to stop evolution?
- ▶ How to choose value for parameter  $X$ ?

## In practice

- ▶ Is my EA working?
- ▶ When to stop evolution?
- ▶ How to choose value for parameter  $X$ ?



On many ( $\geq 30$ ) runs!



# Issues

- ▶ Diversity
- ▶ Variational inheritance
- ▶ Expressiveness . . .

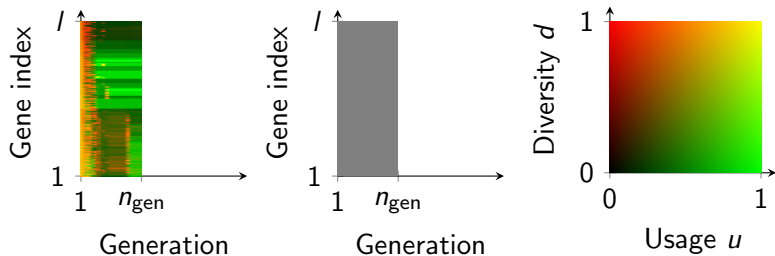
# Diversity

Is the population diverse enough?

- ▶ “No” → too much exploitation → local minimum
- ▶ “Yes” → in principle, no drawbacks
  - ▶ how to measure diversity?
  - ▶ how to enforce/promote diversity?

Giovanni Squillero and Alberto Tonda. “Divergence of character and premature convergence: a survey of methodologies for promoting diversity in evolutionary optimization”. In: *Information Sciences* 329 (2016), pp. 782–799

## Diversity: visualization



Eric Medvet and Tea Tušar. “The DU Map: A Visualization to Gain Insights into Genotype-phenotype Mapping and Diversity”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. Berlin, Germany: ACM, 2017, pp. 1705–1712

# Variational inheritance

Are children similar but not identical to parents?

- ▶ “Too much similar” → too much exploitation → local minimum, no/slow evolution
- ▶ “Too much different” → no exploitation, just coarse exploration (random walk)
- ▶ How to measure? (locality, redundancy, degeneracy, uniformity, ...)
- ▶ How to tackle? Operators, mapping, both?

# Expressiveness

Is the representation (phenotype) expressive enough?

- ▶ “Low expressiveness” → good/optimal solution might not be representable, or might not be reachable
- ▶ “Large expressiveness” → large search space → very long or infinite convergence time

# Fitness landscape

- ▶ How are genotype and fitness spaces related?
- ▶ What does a small step on one correspond to on the other?

**Q:** is phenotype space relevant?

# Genetic Algorithms (GA)

- ▶ Genotype = phenotype = bits string
- ▶  $m = n \approx 1000$ , no overlapping
- ▶ Fitness-proportional selection, or multiobjective (Pareto-based) selection
- ▶ Most widely used/studied
- ▶ Genotypes often encodes numerical parameters

# Genetic Programming (GP)

Focus: individuals are programs

- ▶ Genotype = phenotype = tree (tree-based GP) or list of instructions (linear GP)
- ▶  $m = n \approx 1000$ , overlapping
- ▶ Tournament selection
- ▶ Syntactic/semantic validity?



# Grammatical Evolution (GE)

A form of GP based on GA, given a context-free grammar  $\mathcal{G}$

- ▶ Genotype = bits string, phenotype = string  $\in \mathcal{L}(\mathcal{G})$ , by means of a mapping procedure
- ▶ steady state ( $m \approx 500$ ,  $n = 1$ , overlapping) or  $m = n$ , overlapping
- ▶ Tournament selection

# GE (standard) genotype-phenotype mapping

$g = 01101001\ 00001101\ 01011000\ 00000011\ 11000110\ 01111101$  (bits)  
= 105 13 88 3 198 125 (integers)

$i$	$g_i$	$ r_s $	$j$	$w$	Phenotype $p$
					$\langle \text{expr} \rangle$
0	105	3	0	0	( $\langle \text{expr} \rangle$ $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ )
1	13	3	1	0	( $\langle \text{var} \rangle$ $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ )
2	88	2	0	0	( x $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ )
3	3	4	3	0	( x / $\langle \text{expr} \rangle$ )
4	198	3	0	0	( x / ( $\langle \text{expr} \rangle$ $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ ) )
5	125	3	2	0	( x / ( $\langle \text{num} \rangle$ $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ ) )
0	105	10	5	1	( x / ( 5 $\langle \text{op} \rangle$ $\langle \text{expr} \rangle$ ) )
1	13	4	1	1	( x / ( 5 - $\langle \text{expr} \rangle$ ) )
2	88	3	1	1	( x / ( 5 - $\langle \text{var} \rangle$ ) )
3	3	2	1	1	( x / ( 5 - y ) )

$\langle \text{expr} \rangle ::= ( \langle \text{expr} \rangle \langle \text{op} \rangle \langle \text{expr} \rangle ) \mid \langle \text{var} \rangle \mid \langle \text{num} \rangle$

$\langle \text{op} \rangle ::= + \mid - \mid * \mid /$

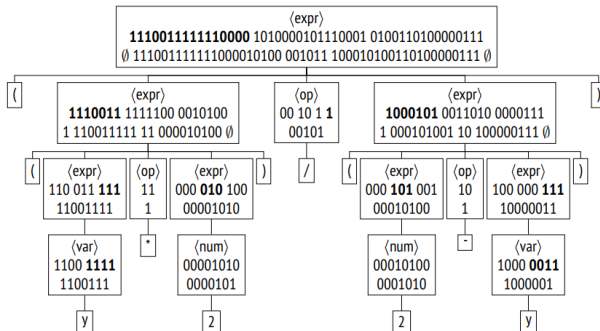
$\langle \text{var} \rangle ::= x \mid y$

$\langle \text{num} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

# An alternative: WHGE genotype-phenotype mapping

$g = 111001111111000010100001011100010100110100000111$

$p = ((y * 2) / (2 - y))$



Eric Medvet. “Hierarchical Grammatical Evolution”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '17. Berlin, Germany: ACM, 2017, pp. 249–250