Data Science for Geosciences

Introduction

2017-18

Organization

Volume

- 5 $\times$ 3h lecture and pratices sessions
- 5 $\times$ 3h Lab/project sessions

Objectives

- model/algorithm analysis for supervised learning
- assess the quality of predictions and inferences
- application of these algorithms on different datasets from geosciences : ecology, geography, ocean-atmosphere, astrophysics, etc. . .

Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9h-12h	General introduction	Regression	Model selection	Deep learning	Project
14h-17h	Project	Classification	Project	Project	Project

Remarks

- ice-breaker on Monday night

Reference books

- Trevor Hastie, Robert Tibshirani et Jerome Friedman (2009)
  The Elements of Statistical Learning (2nd Edition)
  Springer Series in Statistics
- Christopher M. Bishop (2007)
  Pattern Recognition and Machine Learning *Springer*
- Richard O. Duda, Peter E. Hart et David G. Stork (2001)
  Pattern classification (2nd edition) *Wiley*

Supplementary materials, datasets, online courses, ...

- `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`
- `http://research.microsoft.com/en-us/um/people/cmbishop/prml/`
- `https://www.coursera.org/course/ml` *very popular MOOC (Andrew Ng)*
- `https://work.caltech.edu/telecourse.html` *more involved MOOC (Y. Abu-Mostafa)*
- `https://dsg2018.wordpress.com` *webpage for this course*

Data Science

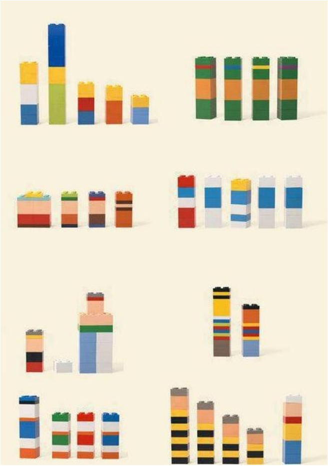
How to extract knowledge or insights from data ?

Learning problems are at the cross-section of several applied fields and science disciplines

- Machine learning arose as a subfield of Artificial Intelligence and Computer Science. Emphasis on large scale implementations and applications (algorithm centered)
- Statistical learning arose as a subfield of Statistics, Applied Maths, Signal Processing,... Emphasizes models and their interpretability (model centered)

☞ There is much overlap : Data Science

Learning problem



Definitions of Learning

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E : data and statistics
- Performance measure P : soptimization
- tasks T : utility
  - automatic translation
  - playing Go
  - ... doing what human does

Definitions of Learning

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E : data and statistics
- Performance measure P : soptimization
- tasks T : utility
  - automatic translation
  - playing Go
  - ... doing what human does

Definitions of Learning

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E : data and statistics
- Performance measure P : soptimization
- tasks T : utility
  - automatic translation
  - playing Go
  - ... doing what human does

Definitions of Learning

Machine Learning in Computer Science

Tom Mitchell (The Discipline of Machine Learning, 2006)

A computer program CP is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E

Key points

- Experience E : data and statistics
- Performance measure P : soptimization
- tasks T : utility
  - automatic translation
  - playing Go
  - ... doing what human does

Experience E : the data !

Type of data : qualitatives / ordinales / quantitatives variables

text	strings
speech	time series
images/videos	2/3d dependences
networks	graphs
games	interaction sequences
	...

Big data (volume, velocity, variety, veracity)

Data are available without having decided to collect them !

- importance of preprocessings (cleaning up, normalization, coding,...)
- importance of a good representation : from raw data to vectors

Objective and performance measures P

Generalize

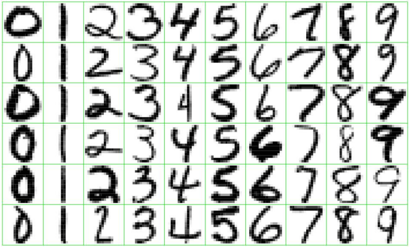
- Perform well (minimize P) on new data (fresh data, i.e. unseen during learning)

☞ Derive good (P/error rate) prediction functions

Examples of Tasks

TODO : To be changed with project applications

Recognition of handwritten digits (US postal envelopes)



- ☞ Predict the class (0,...,9) of each sample from an image of 16×16 pixels, with a pixel intensity coded from 0 to 255
- ▶ Low error rate to avoid wrong allocations of mails !

Supervised classification

Examples of Tasks

TODO : To be changed with project applications

Spams Recognition

Spam

```
WINNING NOTIFICATION
We are pleased to inform you of the result
of the Lottery Winners International
programs held on the 30th january 2005.
[...]  You have been approved for a lump sum
pay out of 175,000.00 euros.
CONGRATULATIONS!!!
```

No Spam

```
Dear George,
Could you please send me the report #1248 on
the project advancement?
Thanks in advance.

Regards,
Cathia
```

☞ Define a model to predict whether an email is spam or not

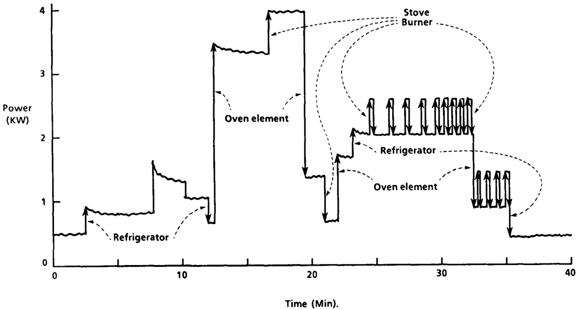
- Low error rate to avoid deleting useful messages, or filling the mailbox with useless emails

supervised classification

Examples of Tasks

TODO : To be changed with project applications

Disaggregation/Prediction of appliance's, or industrial, load



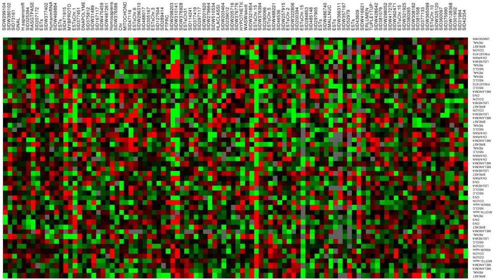
- Individual appliance recognition from load curves
- Predict the energy consumption

supervised or unsupervised classification

Examples of Tasks

TODO : To be changed with project applications

DNA-microarrays



- Genes expression dataset fore several thousand individual genes (columns) and tens of samples (rows)

☞ Classification of genes (resp. samples) with similar expression profiles across samples (resp. genes)

unsupervised classification

Definitions

Variable terminology

- observed data referred to as *input* variables, *predictors* or *features* $\leftarrow$ usually denoted as X
- data to predict referred to as *output* variables, or *responses* $\leftarrow$ usually denoted as Y

Type of prediction problem : regression vs classification

Depending on the type of the *output* variables

- when Y are quantitative data (continuous variables, e.g. electrical load curve values) $\leftarrow$ regression
- when Y are categorical data (discrete qualitative variables, e.g. handwritten digits $Y \in \{0, \dots, 9\}$) $\leftarrow$ classification

Two very close problems

Prediction problem

Assumptions

- couples of input and output variables (X_i, Y_i) are i.i.d.
- input variables X_i are vectors in $\mathbb{R}^p$:

$$X_i = (X_{i,1}, \ldots, X_{i,p})^T \in \mathcal{X} \subset \mathbb{R}^p$$

- output variables Y_i take values :
  - in $\mathcal{Y} \subset \mathbb{R}$ (regression)
  - in a finite set $\mathcal{Y}$ (classification)

Prediction rule

function of prediction / rule of classification $\equiv$ function $f: \ \mathcal{X} \to \mathcal{Y}$ to get predictions

$$\widehat{Y} = f(X)$$

of new elements Y given X

Supervised or unsupervised learning

Training set $\equiv$ available sample $\mathcal{T}$ to learn the prediction rule f

For a sized n training set, different cases :

- Supervised learning : $\mathcal{T} \equiv ((X_1, Y_1), \ldots, (X_n, Y_n))$ input/output couples are available to learn the prediction rule f
- Unsupervised learning : $\mathcal{T} \equiv (X_1, \ldots, X_n)$ only the inputs are available
- Semi-supervised : mixed scenario (often encountered in practice, but less information than in the supervised case)

Academic example of binary classification

- Binary output variables : $Y_i \in \{0, 1\}$,
- Input variables $X_i \in \mathbb{R}^2$, for $i = 1, \ldots, N$

Example of a binary classification problem in $\mathbb{R}^2$. The 2 classes are coded as a binary variable : ORANGE=1, BLUE=0.

Simple linear model for classification

We seek a prediction model based on the linear regression of the outputs $Y \in \{0, 1\}$:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon,$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2)^T$ is a 2D unknown parameter vector

Learning problem $\Leftrightarrow$ Estimation of $\boldsymbol{\beta}$

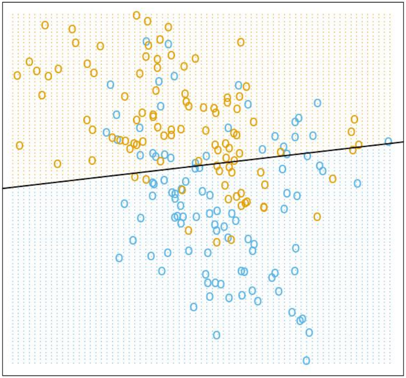
Least Squares Estimator $\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \hat{\beta}_2)^T$: minimize the training error rate (quadratic cost sense)

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (Y_i - \beta_1 X_{i,1} - \beta_2 X_{i,2})^2$$

Classification rule based on least squares regression

$$f(X) = \begin{cases} 1 \text{ if } \widehat{Y} = \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 \geq 0.5, \\ 0 \text{ otherwise} \end{cases}$$

Simple linear model for classification (Cont'd)



Example of classification in $\mathbb{R}^2$*. The 2 classes are coded as a binary variable : ORANGE=1, BLUE=0. The line is the decision boundary* $z = \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0.5$ *: BLUE decision region below, ORANGE one above*

'Black Box' method : k Nearest-Neighbors (k-NN)

The prediction model is directly defined, for $X = x$, as :

$$\widehat{Y}(x) = \frac{1}{k} \sum_{X_i \in N_k(x)} Y_i,$$

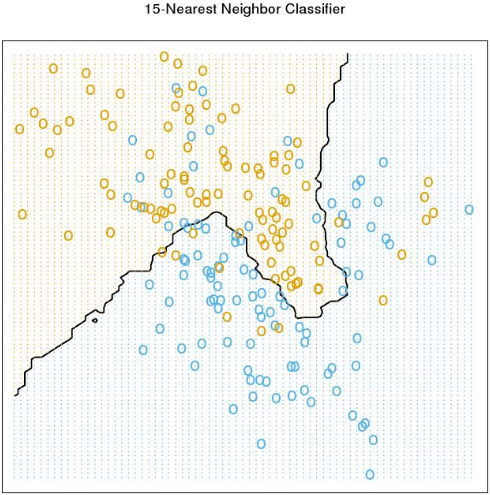
where $N_k(x)$ is the neighborhood of x defined by the k closest inputs X_i in the training set $\{(X_i, Y_i)\}_{i=1\ldots n}$

Classification rule associated with k-NN

$$f(X) = \begin{cases} 1 \text{ if } \widehat{Y}(x) > \frac{1}{2}, \\ 0 \text{ otherwise} \end{cases}$$

$\Leftrightarrow$ majority vote among the k closest neighbors of the testing point x

K Nearest-Neighbors (Cont'd)



Model complexity

Most of methods have a complexity related to their *effective* number of parameters

Linear regression : model order p

E.g. dth degree polynomial regression : $p = d + 1$ parameters a_k s.t.

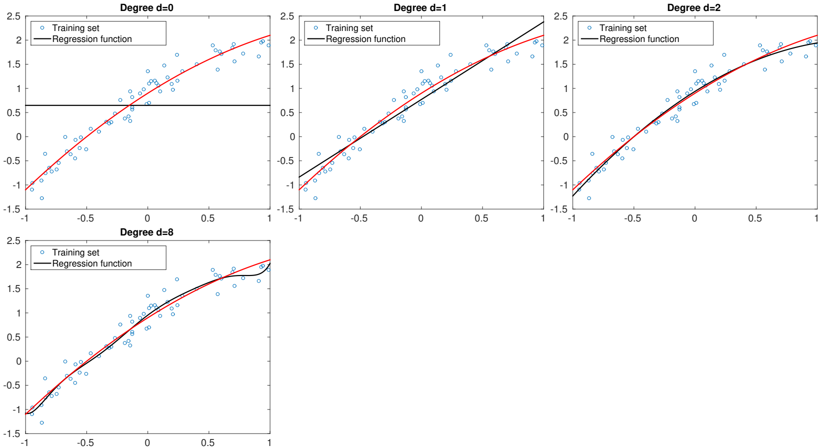
$$\begin{aligned} Y &= \sum_{k=0}^{d} a_k x^k + \epsilon, \\ &= \boldsymbol{X}_d \boldsymbol{a}_d + \epsilon, \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{X}_d &= \left[x^0, x^1, \ldots, x^d \right], \\ \boldsymbol{a}_d &= \left[a_0, a_1, \ldots, a_d \right]^T. \end{aligned}$$

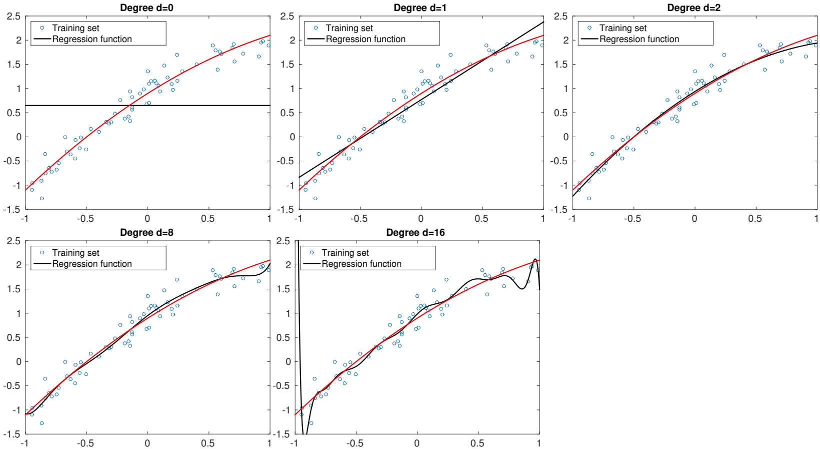
Linear regression : complexity vs stability

Polynomial degree d influence



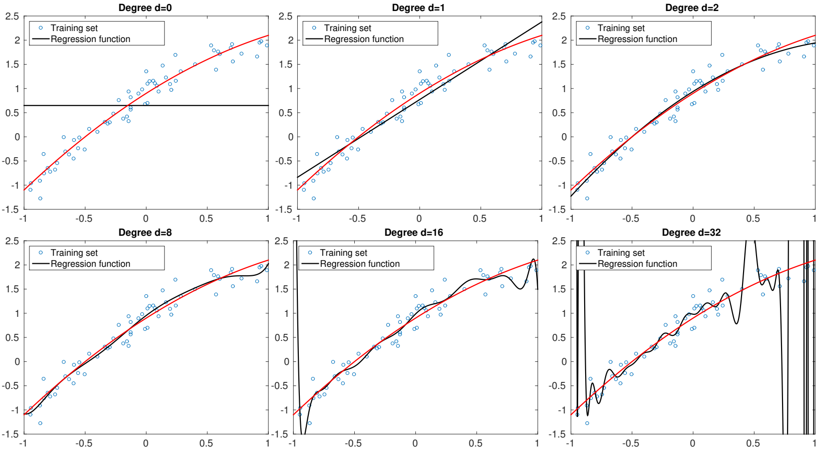
Linear regression : complexity vs stability

Polynomial degree d influence



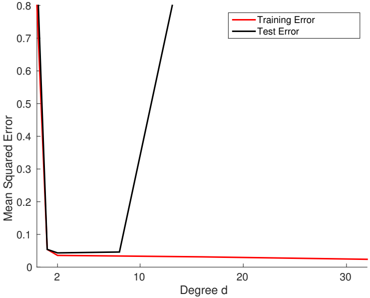
Linear regression : complexity vs stability

Polynomial degree d influence ← over-fitting



Linear Regression : Test error vs Train Error

Error rate vs polynomial order d



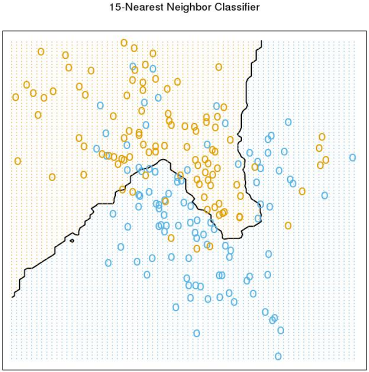
- True error rate (i.e. error rate for test data not used for learning) minimized when $d = 2$ ...
- ... true generative model : order $d = 2$ polynomial (+ white noise)

☞ Training error always decrease with the model complexity. Can't use alone to select the model!

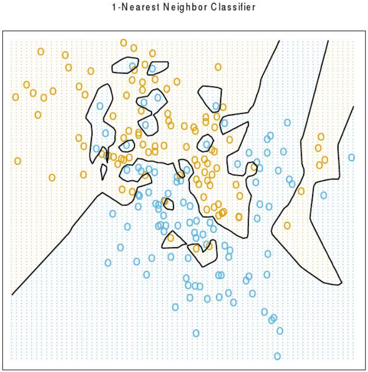
K Nearest-Neighbors

k-NN : complexity parameter k

The effective number of parameters expresses as $N_{\text{eff}} = \frac{N}{k}$, where N is the size of the training sample



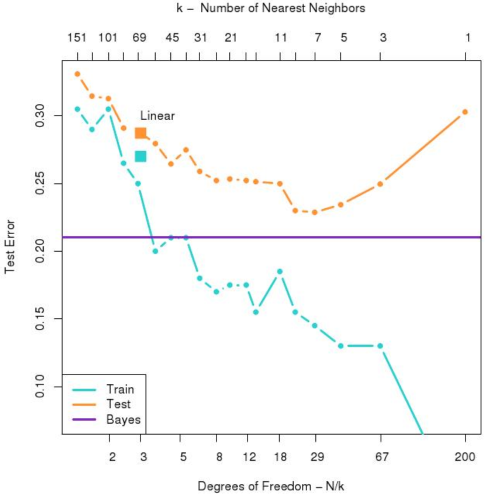
$k = 15,\ N_{\text{eff}} \approx 13$



$k = 1,\ N_{\text{eff}} \approx 200$

- $k = 1 \rightarrow$ training error is 0 !

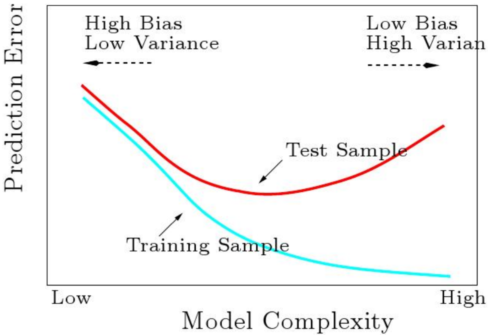
Model Selection



Model Selection (Cont'd)

Fundamental trade-off

- too simple model (high bias) → under-fitting
- too complex model (high variance) → over-fitting



Fundamental Bias-Variance trade-off

if the true model is

$$Y = f(X) + \epsilon,$$

then for any prediction rule $\widehat{f}(X)$, Mean Squared Error (MSE) expresses as

$$E\left[\left(Y - \widehat{f}(x)\right)^2\right] = \text{Var}\left[\widehat{f}(x)\right] + \text{Bias}\left[\widehat{f}(x)\right]^2 + \text{Var}\left[\epsilon\right]$$

- Var $[\epsilon]$ is the *irreducible* part
- as the flexibility of $\widehat{f}$ $\nearrow$, its variance $\nearrow$ and the bias $\searrow$

☞ overfitting/underfitting trade-off

The truth on the example dataset !

Generative model

- For $k = 1, \ldots, 10$, $m_k^1 \sim \mathcal{N}((0,1)^T, \boldsymbol{I})$ and $m_k^0 \sim \mathcal{N}((1,0)^T, \boldsymbol{I})$
- For $l = 1, \ldots, 100$, uniformly pick one m_k^1, then draw $x_l^1 \sim \mathcal{N}(m_k^1, \boldsymbol{I}/5)$
- Same for x_l^0 with the m_k^0 ($N = 200$ for the training sample size)

Bayes Optimal Classifier