INTRODUCTION TO MACHINE LEARNING

Vincent Barra

LIMOS, UMR 6158 CNRS, Clermont-Auvergne University, Clermont-Fd, FRANCE

May 29, 2018

What is Machine Learning

Webster's definition of "to learn"

"Gain knowledge or understanding of, or skill in by study, instruction or experience"

→ Learning a set of new facts

→ Learning HOW to do something

→ Improving ability of something already learned

"Machine Learning"

- Simon [1]: *"Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more effectively the next time"*
- Michalski [2] : *"Learning is constructing or modifying representations of what is being experienced "*
- Mitchell [3] : *"A computer program 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 "*

(1)Simon M- Machine Learning I, 1993, Chapter 2
(2)Michalski R, Carbonell J, Mitchell T (Eds), Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, 1986
(3)Mitchell T, Machine Learning, Chapter 1: Introduction, pp. 1-19, McGraw Hill, 1997.

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

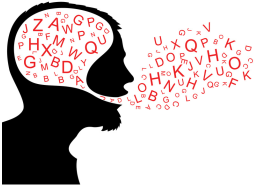
→ Human expertise does not exist
→
→
→
→
→
→

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

→ Human expertise does not exist
→ Humans are unable to explain their expertise
→
→
→
→
→

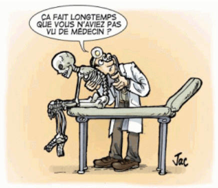


Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

→ Human expertise does not exist
→ Humans are unable to explain their expertise
→ Amount of knowledge is too large for explicit encoding
→
→
→
→



Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

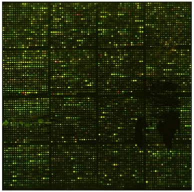
→ Human expertise does not exist
→ Humans are unable to explain their expertise
→ Amount of knowledge is too large for explicit encoding
→ Solution changes in time
→
→
→

Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

- → Human expertise does not exist
- → Humans are unable to explain their expertise
- → Amount of knowledge is too large for explicit encoding
- → Solution changes in time
- → Relationships can be hidden within large amounts of data
- →
- →

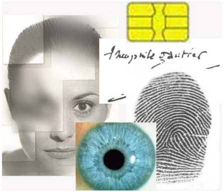


Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

→ Human expertise does not exist
→ Humans are unable to explain their expertise
→ Amount of knowledge is too large for explicit encoding
→ Solution changes in time
→ Relationships can be hidden within large amounts of data
→ Solution needs to be adapted to particular cases
→



Why Learning ?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

Learning is used when

→ Human expertise does not exist
→ Humans are unable to explain their expertise
→ Amount of knowledge is too large for explicit encoding
→ Solution changes in time
→ Relationships can be hidden within large amounts of data
→ Solution needs to be adapted to particular cases
→ New knowledge is constantly being discovered by humans

A simple example

"A computer program 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"

Build a program that learns to detect spams, based on annotated emails

Spam detection

- T detect spams
- E: annotated emails (spams / no spams)
- P: proportion of emails correctly classified

A simple example

"A computer program 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"

Build a program that learns to detect spams, based on annotated emails

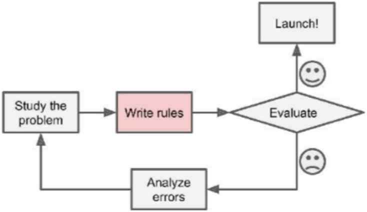
Spam detection

- T detect spams
- E: annotated emails (spams / no spams)
- P: proportion of emails correctly classified

Traditional approach

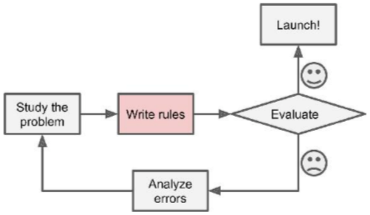
- observe what a spam looks like (frequency of some words, senders,...)
- write a algorithm detecting these patterns
- consider an email as a spam if some patterns are detected
- test and iterate until P is satisfied

A simple example

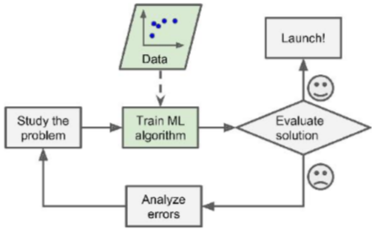


Non trivial task ⇒ huge number of rules / patterns

A simple example

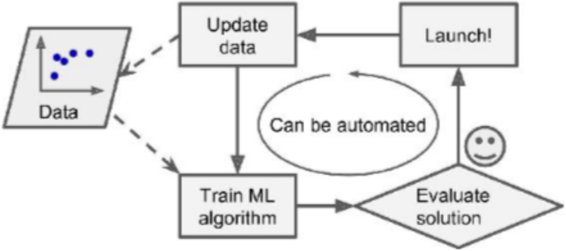


Non trivial task ⇒ huge number of rules / patterns



Machine learning automatically learns what the good features of a spam are.

A simple example



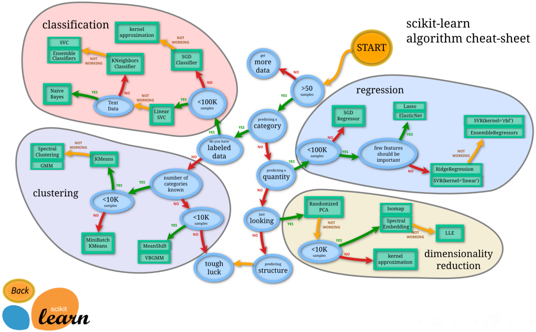
If data / features are changing → Adaptation

Taxonomy

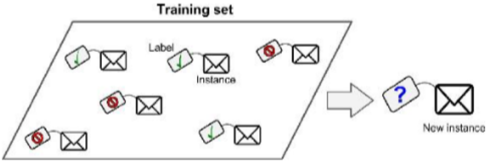
Several criteria

- trained or not: *supervised vs unsupervised vs semi-supervised vs reinforcement learning*
- trained gradually with the data or not: *online vs batch*
- based on known examples or built predictive models: *instance-based vs model-based.*
- objective: *regression vs. classification*

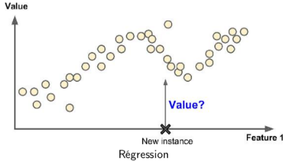
Non exhaustive and combinable.



Supervised learning



Classification



Regression

Supervised learning: a special focus

Focus on supervised learning:

- Viewed from a statistical point of view
- Help to undersand the underlying notions (model, over/under fitting...)
- Relations with several other notions (optimization,...)

Supervised learning from a statistical point of view

Vapnik definition of a learning model

- A random vector generator G giving $x \in \mathbb{R}^n$ i.i.d. using fixed but unknown $P(x)$
- A supervisor S giving for each input x a value y using a conditional fixed but unknown distribution$P(y|x)$
- A learning machine LM implementing a set of functions $\mathcal{F}$

SUPERVISED LEARNING FROM A STATISTICAL POINT OF VIEW

Statistical learning problem $\Leftrightarrow$choose f in $\mathcal{F}$ that best models S

LEARNING (OR TRAINING) SET

Choice of $f \Rightarrow$ **training set** $\{(x_1, y_1), \ldots, (x_l, y_l)\}$: l iid observations using $P(x, y) = P(x) P(y|x)$.

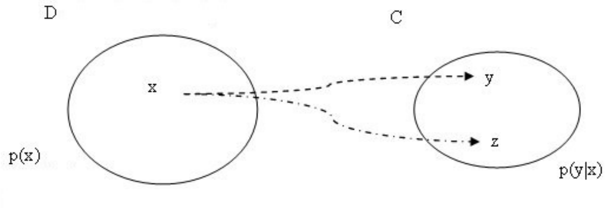
Supervised learning from a statistical point of view

Statistical learning problem $\Leftrightarrow$choose f in $\mathcal{F}$ that best models S

Learning (or training) set

Choice of $f \Rightarrow$ **training set** $\{(x_1, y_1), \ldots, (x_l, y_l)\}$: l iid observations using $P(x, y) = P(x)\,P(y|x)$.

Supervised learning from a statistical point of view



$S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$ drawn using $p(x, y) = p(x)\, p(y|x)$
Objective: Find $f : D \to C$ with minimal error $R(f) = P(y \neq f(x))$.

Supervised learning from a statistical point of view

Remarks

- $\mathbb{R}^n$ is continuous
- Non deterministic model
  - non deterministic target problem ;
  - noisy problem;
  - $\mathbb{R}^n$ only partially describes a complex situation.
- Searching for a deterministic solution.
- non parametric model $\Rightarrow$ no constraint on $\mathcal{F}$.

Supervised learning from a statistical point of view

Remarks

- $\mathbb{R}^n$ is continuous
- Non deterministic model
  - non deterministic target problem ;
  - noisy problem;
  - $\mathbb{R}^n$ only partially describes a complex situation.
- Searching for a deterministic solution.
- non parametric model $\Rightarrow$ no constraint on $\mathcal{F}$.

Supervised learning from a statistical point of view

Remarks

- $\mathbb{R}^n$ is continuous
- Non deterministic model
  - non deterministic target problem ;
  - noisy problem;
  - $\mathbb{R}^n$ only partially describes a complex situation.
- Searching for a deterministic solution.
- non parametric model $\Rightarrow$ no constraint on $\mathcal{F}$.

Supervised learning from a statistical point of view

Remarks

- $\mathbb{R}^n$ is continuous
- Non deterministic model
  - non deterministic target problem ;
  - noisy problem;
  - $\mathbb{R}^n$ only partially describes a complex situation.
- Searching for a deterministic solution.
- non parametric model $\Rightarrow$ no constraint on $\mathcal{F}$.

Supervised learning from a statistical point of view

Loss function

$$L(y, f(x)) = \mathbb{1}_{y \neq f(x)}$$

Measures the difference between S (y) and LM $(f(x))$

Risk or error

$$R(f) = \int L(y, f(x))\, dP(x, y) = P(y \neq f(x))$$

$\Rightarrow$ Expected value of the loss function= probability that f predicts a value different from S.

Supervised learning from a statistical point of view

Learning issue $\Leftrightarrow$ Knowing a training set, find $f \in \mathcal{F}$ minimizing $R(f)$.

Extensions

This formulation can be extended to regression and density estimation problems, e.g.:

- $L(y, f(x)) = (y - f(x))^2$
- $L(y, f(x)) = -log(f(x))$

Supervised learning from a statistical point of view

Learning issue $\Leftrightarrow$ Knowing a training set, find $f \in \mathcal{F}$ minimizing $R(f)$.

Extensions

This formulation can be extended to regression and density estimation problems, e.g.:

- $L(y, f(x)) = (y - f(x))^2$
- $L(y, f(x)) = -log(f(x))$

Supervised learning from a statistical point of view

For classification, $\exists$ a function with minimal risk (Bayes' decision rule)

$$f_{Bayes}(x) = argmax_y P(y|x)$$

f_{Bayes} : ideal function

Learning issue $\Leftrightarrow$ Knowing a training set, find $f \in \mathcal{F}$ as close as possible to f_{Bayes}

Supervised learning from a statistical point of view

For classification, $\exists$ a function with minimal risk (Bayes' decision rule)

$$f_{Bayes}(x) = argmax_y P(y|x)$$

f_{Bayes} : ideal function

Learning issue $\Leftrightarrow$ Knowing a training set, find $f \in \mathcal{F}$ as close as possible to f_{Bayes}

Supervised learning from a statistical point of view

Let suppose there exists $f_{opt} \in \mathcal{F}$ with minimum risk:

$$0 \leq R\left(f_{Bayes}\right) \leq R\left(f_{opt}\right) = \underbrace{R\left(f_{Bayes}\right)}_{\text{non-deterministic}} + \underbrace{\left(R\left(f_{opt}\right) - R\left(f_{Bayes}\right)\right)}_{\text{structural error}}$$

Use expressive $\mathcal{F}$ spaces to allow:

- the best function to be close to f_{Bayes}
- functions to be sufficiently handy

Supervised learning from a statistical point of view

Let suppose there exists $f_{opt} \in \mathcal{F}$ with minimum risk:

$$0 \leq R\left(f_{Bayes}\right) \leq R\left(f_{opt}\right) = \underbrace{R\left(f_{Bayes}\right)}_{\text{non-deterministic}} + \underbrace{\left(R\left(f_{opt}\right) - R\left(f_{Bayes}\right)\right)}_{\text{structural error}}$$

Use expressive $\mathcal{F}$ spaces to allow:

- the best function to be close to f_{Bayes}
- functions to be sufficiently handy

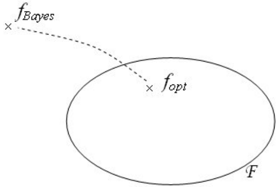
SUPERVISED LEARNING FROM A STATISTICAL POINT OF VIEW

Let suppose there exists $f_{opt} \in \mathcal{F}$ with minimum risk:

$$0 \leq R\left(f_{Bayes}\right) \leq R\left(f_{opt}\right) = \underbrace{R\left(f_{Bayes}\right)}_{\text{non-deterministic}} + \underbrace{\left(R\left(f_{opt}\right) - R\left(f_{Bayes}\right)\right)}_{\text{structural error}}$$

Use expressive $\mathcal{F}$ spaces to allow:

- the best function to be close to f_{Bayes}
- functions to be sufficiently handy



Supervised learning from a statistical point of view

Natural idea: select $f \in \mathcal{F}$ best classifying the training set

Empirical risk

Empirical risk of f on $\{(x_1, y_1), \ldots, (x_l, y_l)\}$

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)) = \frac{Card\{i | f(x_i) \neq y_i\}}{l}$$

Empirical Risk Minimization (ERM)

Find $f \in \mathcal{F}$ (f_{emp}) minimizing $R_{emp}(f)$

$$R(f_{emp}) = R(f_{Bayes}) + (R(f_{opt}) - R(f_{Bayes})) + (R(f_{emp}) - R(f_{opt}))$$

Supervised learning from a statistical point of view

Natural idea: select $f \in \mathcal{F}$ best classifying the training set

Empirical risk

Empirical risk of f on $\{(x_1, y_1), \ldots, (x_l, y_l)\}$

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)) = \frac{Card\{i | f(x_i) \neq y_i\}}{l}$$

Empirical Risk Minimization (ERM)

Find $f \in \mathcal{F}$ (f_{emp}) minimizing $R_{emp}(f)$

$$R(f_{emp}) = R(f_{Bayes}) + (R(f_{opt}) - R(f_{Bayes})) + (R(f_{emp}) - R(f_{opt}))$$

Supervised learning from a statistical point of view

Natural idea: select $f \in \mathcal{F}$ best classifying the training set

Empirical risk

Empirical risk of f on $\{(x_1, y_1), \ldots, (x_l, y_l)\}$

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} L(y_i, f(x_i)) = \frac{Card\{i | f(x_i) \neq y_i\}}{l}$$

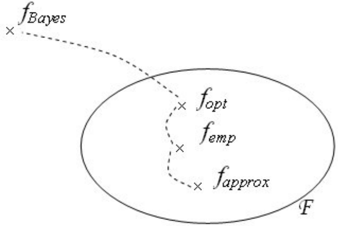
Empirical Risk Minimization (ERM)

Find $f \in \mathcal{F}$ (f_{emp}) minimizing $R_{emp}(f)$

$$R(f_{emp}) = R(f_{Bayes}) + (R(f_{opt}) - R(f_{Bayes})) + (R(f_{emp}) - R(f_{opt}))$$

Supervised learning from a statistical point of view

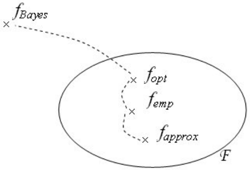
In practice, impossible to compute f_{emp} in reasonable time $\Rightarrow f_{approx} \approx f_{emp}$.



Supervised learning from a statistical point of view

At least 3 reasons hinder the results of a ML algorithm:

- *weak expressivity of $\mathcal{F}$* : structural error;
- *Unconsistency or the ERM principle* : do we get close to f_{opt} with the training set (and its number of examples) ?
- *Difficulty to minimize the empirical risk.*



Supervised learning from a statistical point of view

- ERM does not allow to be close to the optimal function in all cases.
  $\Rightarrow$ the training set is by nature stochastic
- $\mathcal{F}$ too rich $\Rightarrow$ ERM can overfit.

Supervised learning from a statistical point of view

- ERM does not allow to be close to the optimal function in all cases.
  $\Rightarrow$ the training set is by nature stochastic
- $\mathcal{F}$ too rich $\Rightarrow$ ERM can overfit.

Supervised learning from a statistical point of view

Serious problem

- f_{opt} close to f_{bayes} needs a rich $\mathcal{F}$;
- Find f_{opt} using ERM ned not so rich $\mathcal{F}$.

Extreme examples:

- $\mathcal{F} = \{f_0\}$, $f_{opt} = f_0$ but does not mimimize R_{emp};
- $\mathcal{F}$= all possible functions, $f_{bayes} \in \mathcal{F}$ but also all functions minimizing R_{emp} including $f_{byheart}$.

Bias-variance tradeoff

Bias $\approx$ distance between f_{bayes} and f_{opt}
Variance $\approx$ distance between f_{opt} and f_{emp}

Supervised learning from a statistical point of view

Serious problem

- f_{opt} close to f_{bayes} needs a rich $\mathcal{F}$;
- Find f_{opt} using ERM ned not so rich $\mathcal{F}$.

Extreme examples:

- $\mathcal{F} = \{f_0\}$, $f_{opt} = f_0$ but does not mimimize R_{emp};
- $\mathcal{F}$= all possible functions, $f_{bayes} \in \mathcal{F}$ but also all functions minimizing R_{emp} including $f_{byheart}$.

Bias-variance tradeoff

Bias $\approx$ distance between f_{bayes} and f_{opt}
Variance $\approx$ distance between f_{opt} and f_{emp}

Supervised learning from a statistical point of view

Serious problem

- f_{opt} close to f_{bayes} needs a rich $\mathcal{F}$;
- Find f_{opt} using ERM ned not so rich $\mathcal{F}$.

Extreme examples:

- $\mathcal{F} = \{f_0\}$, $f_{opt} = f_0$ but does not mimimize R_{emp};
- $\mathcal{F}$= all possible functions, $f_{bayes} \in \mathcal{F}$ but also all functions minimizing R_{emp} including $f_{byheart}$.

Bias-variance tradeoff

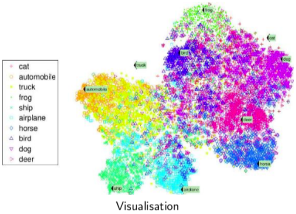
Bias $\approx$ distance between f_{bayes} and f_{opt}
Variance $\approx$ distance between f_{opt} and f_{emp}

Supervised learning

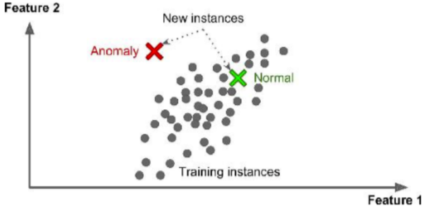
Main algorithms

- k-nearest neighbors
- Linear regression
- Logistic regression
- SVM, SVR
- Decision trees and random forests
- Shallow and deep neural networks

Unsupervised learning



Clustering



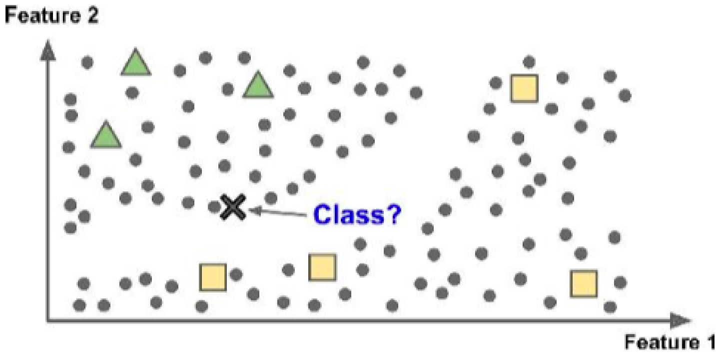
Anomaly detection

Unsupervised learning

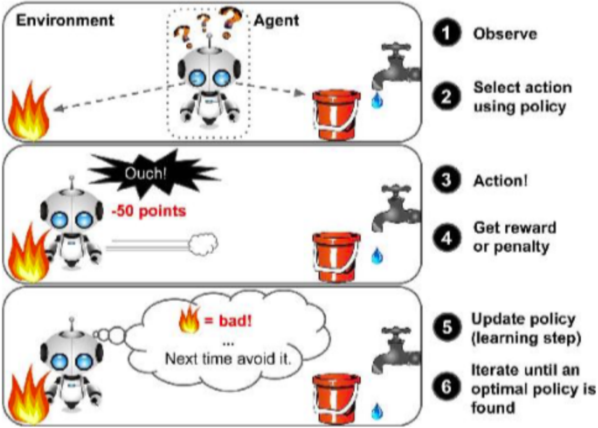
Main algorithms

- Clustering
  - k-means and fuzzy variations
  - Hierarchical cluster analysis
  - EM
- Visualisation and dimension reduction
  - PCA, ICA
  - Non linear techniques: ISOMAP, LLE,...
  - Kernel methods
  - t-SNE
- Association rules

Semi supervised learning



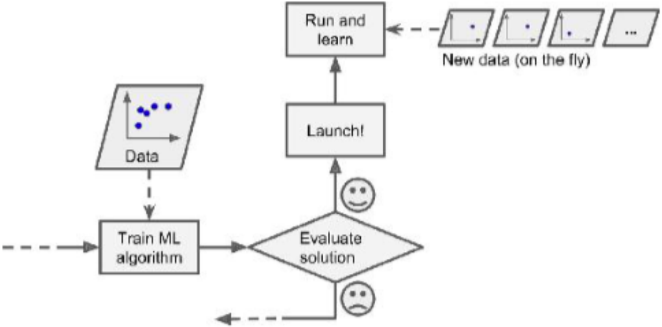
Reinforcement learning



Batch / Inline learning

Does the ML algorithm have the ability to incrementally update, following a data stream ?

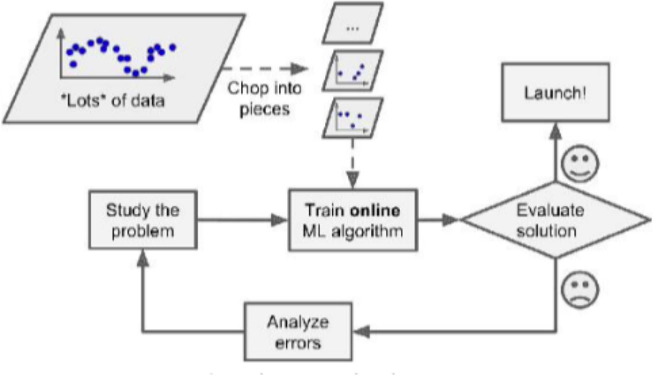
Inline learning



Inline... misleading term → incremental & offline learning

Inline learning

Huge databases



Inline learning

Learning rate

How fast an inline ML algorithm has to adapt to new data (and then forget the older ones) ?

$\Rightarrow$ Define a learning rate:

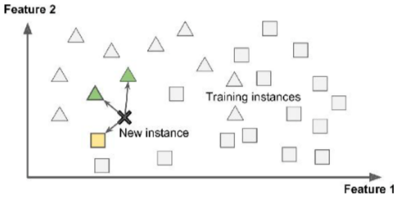
- too fast: unstable system, too sensitive to erroneous data
- too slow: the algorithm will not be able to adapt

Generalization

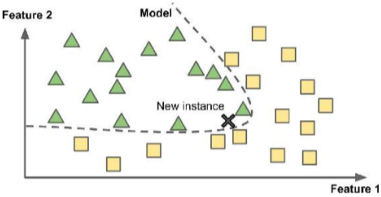
Generalization

Capacity of an algorithm to correctly predict on new data.
Two main approaches:

- instance-based (without a model)
- model-based



instance-based



model-based

Example

Simple example: construction of a model on simple data

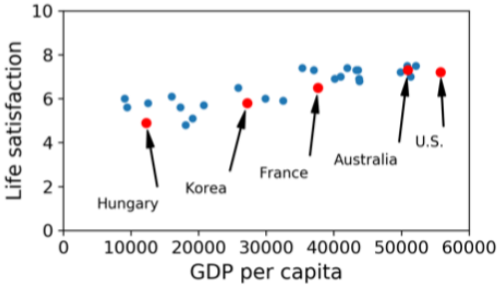
Data

- "Better life" data, OCDE
- income distribution by country and subjective feelings (happyness)

Country	Income (USD)	Happyness
Hungary	12240	4.9
South Korea	27195	5.8
France	37675	6.5
Australia	50962	7.3
U.S.	55805	7.2

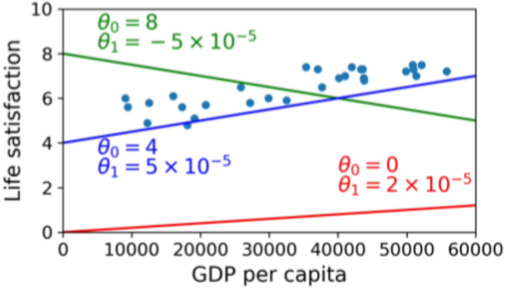
Can money buy happyness ?

Example



Any tendancy ?

Example



$$\text{happyness} = \theta_0 + \theta_1 \text{income}$$

Example

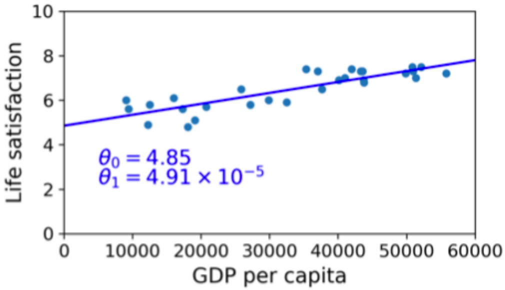
Optimal parameter values

Optimal in the sens of a performance measure

- utility function: measures to which extent the model performs well
- cost function: measures to which extent the model is bad

Linear regression case

In general: cost function measuring the distance between predictions and real values on the training set.



Main challenges of ML

Two main problems:

- A wrong algorithm
- bad, missing, noisy and/or too few data

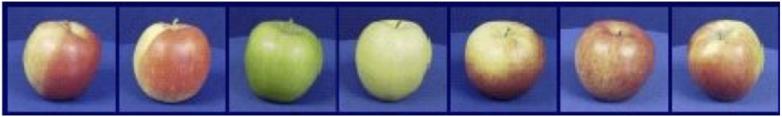
Too few data

The child

To learn what an apple is, only have to show (and repeat) an apple, and pronounce the word. The child is then able to recognize all varieties of apples, whatever the shape and color

The machine

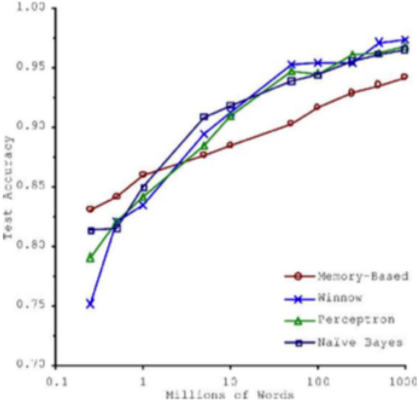
A lot of data is necessary to learn the concept.
Even for simple problems, thousands of examples needed.



Too few data

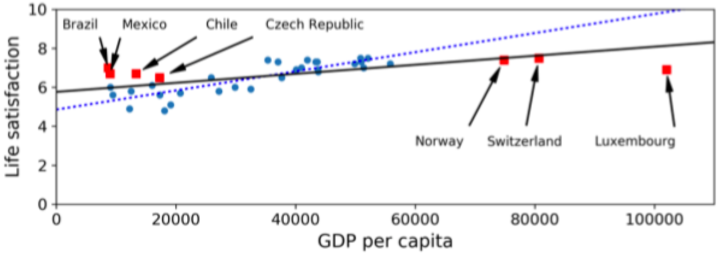
For best performances: simple (and even naive) algorithm and huge amount of data.

Example: performance of simple algorithms on a difficult problem (desambiguation of “too”, “two” or “to”)



Non representative data

For generalization purposes, training data must be representative of future data.



Poor data

Clean the data

- if points are clearly outliers, remove or manually correct them
- if some values (attributes) are missing for some data:
  - ignore the corresponding attribute
  - ignore the corresponding data
  - fill the missing values (mean, median...)
  - learn several models combining these approaches

Poor data

Clean the data

- if points are clearly outliers, remove or manually correct them
- if some values (attributes) are missing for some data:
  - ignore the corresponding attribute
  - ignore the corresponding data
  - fill the missing values (mean, median...)
  - learn several models combining these approaches

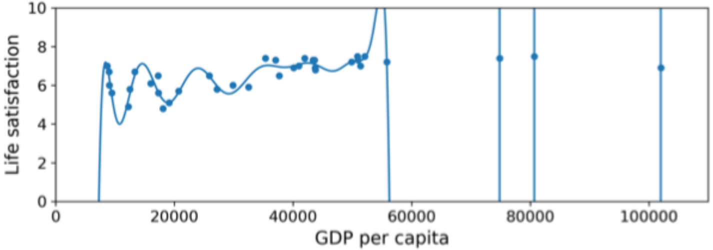
Filter the attributes

- variable selection
- variable extraction
- creation of new attributes from new data.

Overfitting

Overfitting

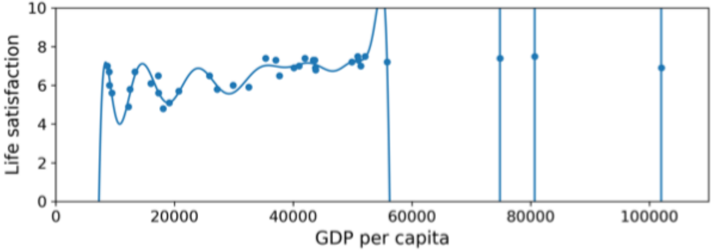
The algorithm fits very well the training set but behaves poorly on generalization



Overfitting

Overfitting

The algorithm fits very well the training set but behaves poorly on generalization



Why ?

Model too complex w.r.t. noise level and.or number of data

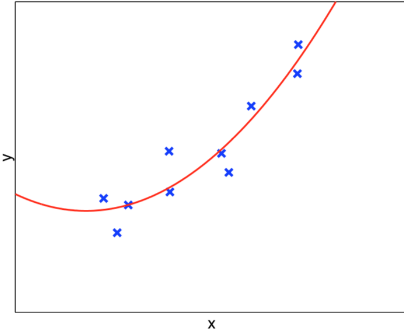
- simplify the model
- use more data
- reduce the amount of noise in the data

Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 2

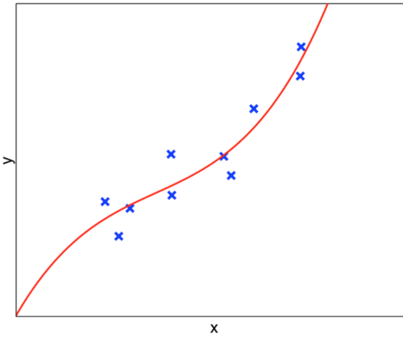


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 3

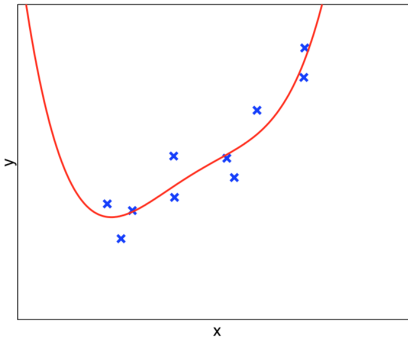


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 4

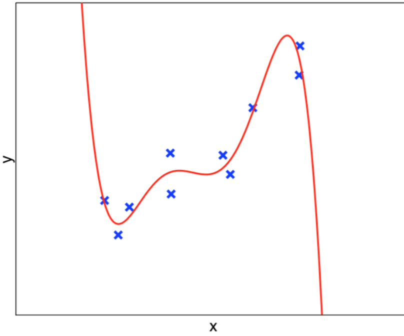


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 5

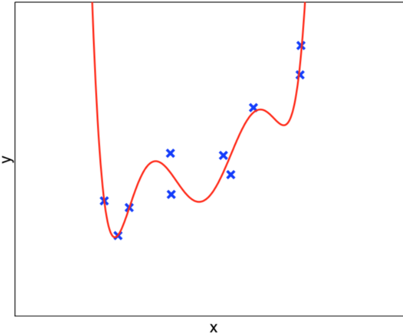


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 6

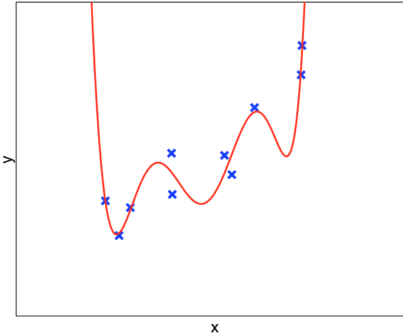


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 7

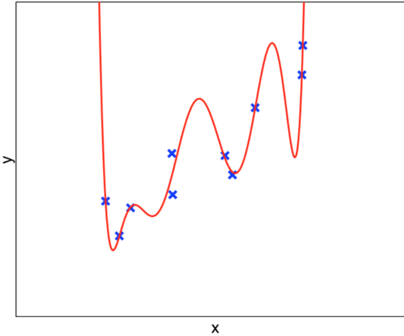


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

Order 8

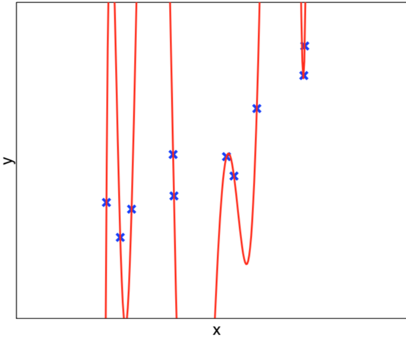


Overfitting

A visual example of overfitting

Polynomial interpolation of a set of points

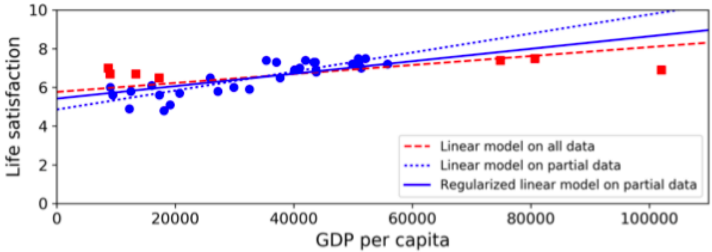
Order 9



Regularization

Definition

Constrain the model to simplify it.

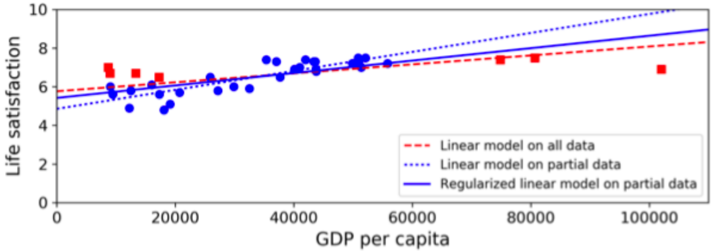


simple example: give a limit for θ_1

Regularization

Definition

Constrain the model to simplify it.



simple example: give a limit for θ_1

Hyperparameter

The quantity of regularization is controled by an hyperparameter (learning parameter)

- a prori fixed during the training phase
- the higher the hyperparameter, the more constrained the model will be
- hyperparameter(s) need(s) to be tuned (important issue)

Underfitting

Definition

The model is too simple:

- choose a more complex model (with more parameters)
- find better attributes
- lower the regularization

Test set

Once the model is learned, one has to evaluate it and if necessary tune it.

Training/test sets

The only way to see if the model generalizes well is to test it on new data

- A subset of the initial set ($\approx$ 80%)will serve as a learning set $\rightarrow$ training error
- the rest will serve as a test set $\rightarrow$ generalization error

Test set

Once the model is learned, one has to evaluate it and if necessary tune it.

Training/test sets

The only way to see if the model generalizes well is to test it on new data

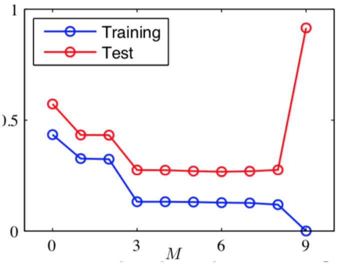
- A subset of the initial set ($\approx$ 80%)will serve as a learning set $\rightarrow$ training error
- the rest will serve as a test set $\rightarrow$ generalization error

Overfitting

generalization error$>$ training error $\Rightarrow$ Overfitting

Test set

- The training error decreases with the complexity of the model
- The generalization error decreases at first, then starts increasing
- As we will see, cross-validation will help:
  - Find a good model, using a validation set
  - Report unbiased results, using a test set, untouched during either parameter training or validation



Validation set

Hyperparameter tuning

When comparing several models (using different values for the hyperparameter(s)) on the test set, this test will be "learned"

Validation set

Hyperparameter tuning

When comparing several models (using different values for the hyperparameter(s)) on the test set, this test will be "learned"

Learning/Validation sets

Learning set / Validation set
The models are tested on the validation set, the best is retained. Then this model is applied on the test set to evaluate it.

Cross validation

Risk: learn the validation set

Principle

- divide the learning set in v subsets
- learn the modl with $v-1$ subsets
- test using the last subset
- repeat v times, using each of the v subsets as a test set

Final error: mean of the v learning errors

Cross validation

Risk: learn the validation set

Principle

- divide the learning set in v subsets
- learn the modl with $v-1$ subsets
- test using the last subset
- repeat v times, using each of the v subsets as a test set

Final error: mean of the v learning errors

Leave one out

v : number of examples

Performance measure

Measuring the performance of a classifier is generally harder than for a regression algorithm.

- cross validation (can be difficult if the classes are non equilibrated)
- confusion matrix (binary and multiclass cases)

Performance measure

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Example: binary confusion matrix C

- $C_{1,1}$: true positives (TP)
- $C_{2,2}$: true negatives (TN)
- $C_{1,2}$: false positives (FP)
- $C_{2,1}$: true negatives (FN)

Performance measure

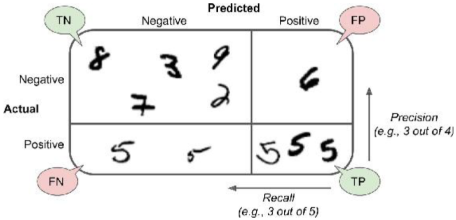
$$C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

Example: binary confusion matrix C

- $C_{1,1}$: true positives (TP)
- $C_{2,2}$: true negatives (TN)
- $C_{1,2}$: false positives (FP)
- $C_{2,1}$: true negatives (FN)

Precision / Recall

- precision $P = \frac{TP}{TP+FP}$:
- recall $\frac{TP}{TP+FN}$



Performance measure

F_1 score

$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Performance measure

F_1 score

$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Harmonic mean

Good performances for classifiers with similar P and R values

Performance measure

F_1 score

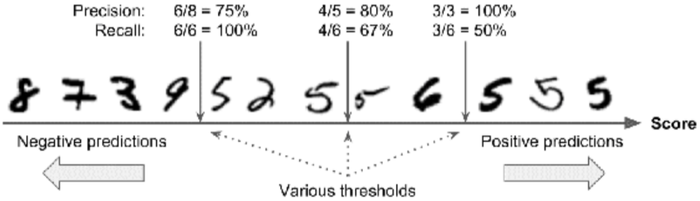
$$F_1 = 2\frac{P.R}{P+R} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Harmonic mean

Good performances for classifiers with similar P and R values

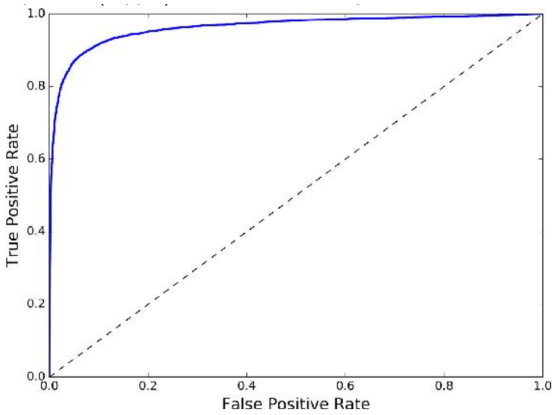
P/R compromise

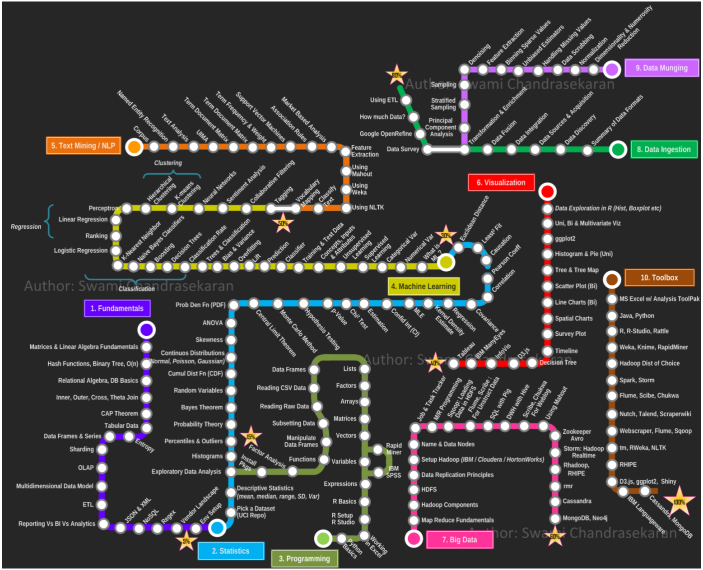
- In general, improving P lowers R and vice versa.
- Decision function, returning a value compared to a threshold



Performance measure

ROC (Receiver Operating Characteristic): TP rate vs. FP rate





1. Fundamentals
Matrices & Linear Algebra Fundamentals
Hash Functions, Binary Tree, O(n)
Relational Algebra, DB Basics
Inner, Outer, Cross, Theta Join
CAP Theorem
Tabular Data
Data Frames & Series
Entropy
Sharding
OLAP
Multidimensional Data Model
ETL
Reporting Vs BI Vs Analytics
JSON & XML
NoSQL
Regex
Vendor Landscape
Env Setup
2. Statistics
Pick a Dataset (UCI Repo)
Descriptive Statistics (mean, median, range, SD, Var)
Exploratory Data Analysis
Histograms
Percentiles & Outliers
Probability Theory
Bayes Theorem
Random Variables
Cumul Dist Fn (CDF)
Continuos Distributions (Normal, Poisson, Gaussian)
Skewness
ANOVA
Prob Den Fn (PDF)
Central Limit Theorem
Monte Carlo Method
Hypothesis Testing
p-Value
Chi² Test
Estimation
Confid Int (CI)
MLE
Kernel Density Estimate
Regression
Covariance
Correlation
Pearson Coeff
Causation
Least² Fit
Euclidean Distance
3. Programming
Python Basics
Working in Excel
R Setup R Studio
R Basics
Expressions
Variables
IBM SPSS
Rapid Miner
Vectors
Matrices
Arrays
Factors
Lists
Data Frames
Reading CSV Data
Reading Raw Data
Subsetting Data
Manipulate Data Frames
Functions
Install Pkgs
Factor Analysis
4. Machine Learning
What is ML?
Numerical Var
Categorical Var
Supervised Learning
Unsupervised Learning
Concepts, Inputs & Attributes
Training & Test Data
Classifier
Prediction
Lift
Overfitting
Bias & Variance
Trees & Classification
Classification Rate
Decision Trees
Boosting
Naive Bayes Classifiers
K-Nearest Neighbor
Logistic Regression
Ranking
Linear Regression
Perceptron
Regression
Classification
Hierarchical Clustering
K-means Clustering
Clustering
Neural Networks
Sentiment Analysis
Collaborative Filtering
Tagging
5. Text Mining / NLP
Corpus
Named Entity Recognition
Text Analysis
UIMA
Term Document Matrix
Term Frequency & Weight
Support Vector Machines
Association Rules
Market Based Analysis
Feature Extraction
Using Mahout
Using Weka
Using NLTK
Classify Text
Vocabulary Mapping
6. Visualization
Data Exploration in R (Hist, Boxplot etc)
Uni, Bi & Multivariate Viz
ggplot2
Histogram & Pie (Uni)
Tree & Tree Map
Scatter Plot (Bi)
Line Charts (Bi)
Spatial Charts
Survey Plot
Timeline
Decision Tree
D3.js
InfoVis
IBM ManyEyes
Tableau
7. Big Data
Map Reduce Fundamentals
Hadoop Components
HDFS
Data Replication Principles
Setup Hadoop (IBM / Cloudera / HortonWorks)
Name & Data Nodes
Job & Task Tracker
M/R Programming
Sqoop: Loading Data in HDFS
Flume, Scribe: For Unstruct Data
SQL with Pig
DWH with Hive
Scribe, Chukwa For Weblog
Using Mahout
Zookeeper Avro
Storm: Hadoop Realtime
Rhadoop, RHIPE
rmr
Cassandra
MongoDB, Neo4j
8. Data Ingestion
Summary of Data Formats
Data Discovery
Data Sources & Acquisition
Data Integration
Data Fusion
Transformation & Enrichment
Data Survey
Google OpenRefine
How much Data?
Using ETL
9. Data Munging
Dimensionality & Numerosity Reduction
Normalization
Data Scrubbing
Handling Missing Values
Unbiased Estimators
Binning Sparse Values
Feature Extraction
Denoising
Sampling
Stratified Sampling
Principal Component Analysis
10. Toolbox
MS Excel w/ Analysis ToolPak
Java, Python
R, R-Studio, Rattle
Weka, Knime, RapidMiner
Hadoop Dist of Choice
Spark, Storm
Flume, Scibe, Chukwa
Nutch, Talend, Scraperwiki
Webscraper, Flume, Sqoop
tm, RWeka, NLTK
RHIPE
D3.js, ggplot2, Shiny
IBM Languageware
Cassandra, MongoDB
5%
15%
30%
40%
50%
60%
80%
100%
Author: Swami Chandrasekaran