Apprentissage Artificiel
(Statistical Machine-Learning)
General framework + Supervised Learning

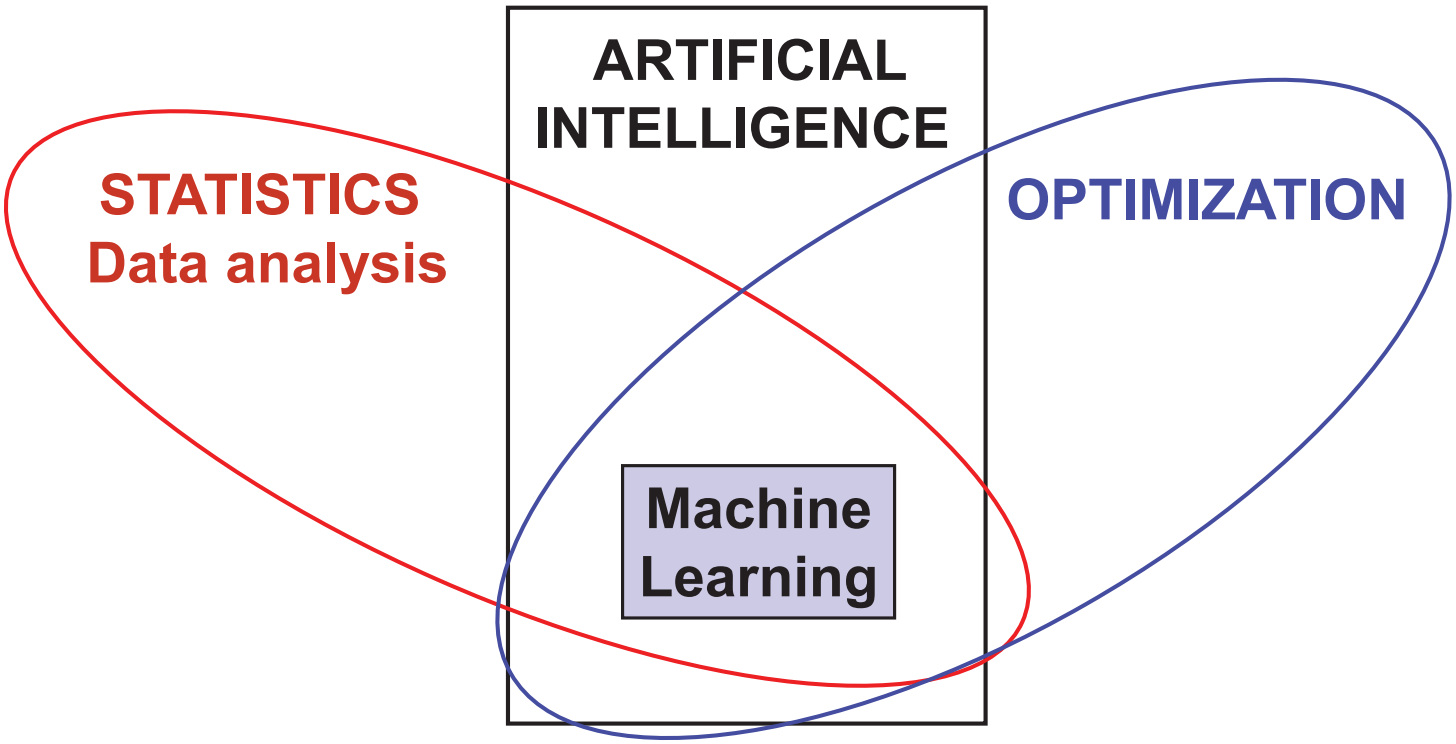
Pr. Fabien MOUTARDE
Center for Robotics
MINES ParisTech
PSL Université Paris

Fabien.Moutarde@mines-paristech.fr
http://people.mines-paristech.fr/fabien.moutarde

Outline

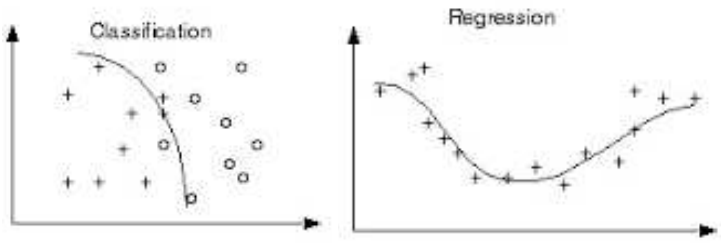
- **Intro: What is Statistical Machine-Learning?**
- Typology of Machine-Learning
- General formalism for SUPERVISED Learning
- Evaluating learnt models:
  metrics for CLASSIFICATION
- Generalization *vs.* overfitting

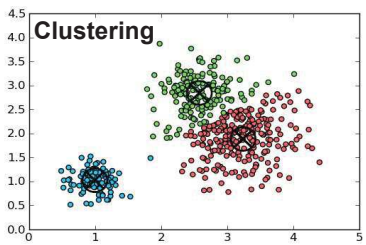
What is Statistical Machine-Learning?

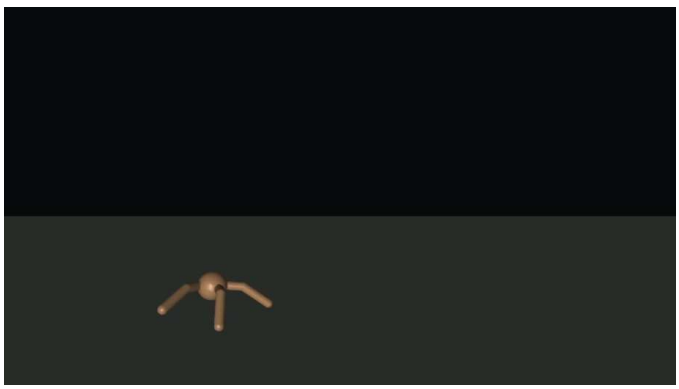


Statistical Machine-Learning

- One of many sub-fields of Artificial Intelligence
- Application of *optimization methods* to statistical modelling
- Data-driven *mathematical* modelling, for automated *classification*, *regression*, *partitioning/clustering*, or *decision/behavior rule*

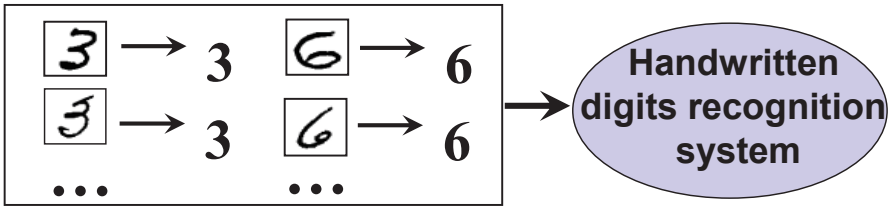






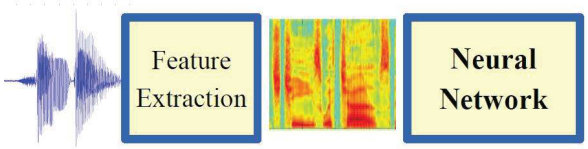
Real-world examples of Machine-Learning applications

- **Handwritten characters recognition**



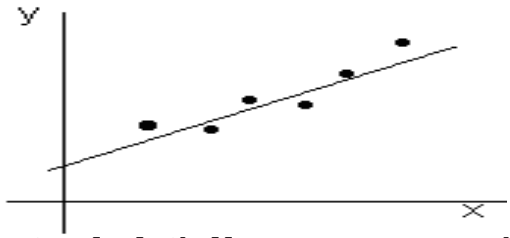
- **Object category visual recognition**

- **Speech recognition**



- Multi-factorial forecasting
- Natural Language understanding
- Playing GO!
- MANY MANY MORE…

One of simplest ML algorithm: Least Squares Linear Regression



- **Model:** (straight) line $y=ax+b$ (2 parameters a and b)
- **Data:** n points with target value $(x_i,y_i)\in\Re^2$
- **Cost function:** sum of squares of deviation from line

$$K=\Sigma_i (y_i-a.x_i-b)^2$$

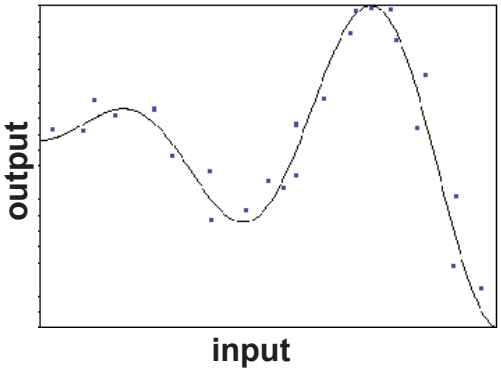
- **Algorithm:** direct (or iterative) solving of linear system

$$\begin{pmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{pmatrix}$$

[Question: Where does this equation come from?]

Regression *vs.* classification

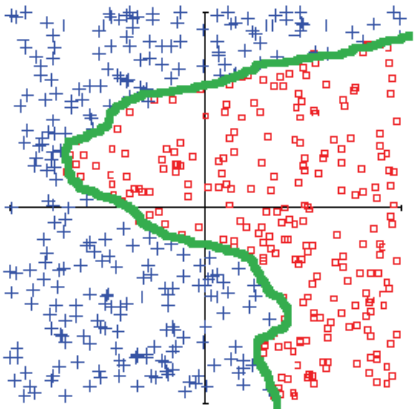
Regression



points = examples ➔ curve = regression

***Continuous* output(s)**

Classification



Input = point position

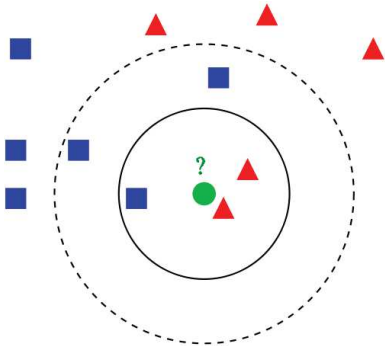
target Output = class label (□ =-1,+=+1)

⬇

Function label=f(x) (and separation boundary)

***Discrete* output(s)**

Simplest *classification* method: Nearest Neighbors algorithm



Principle of Nearest Neighbors (kNN) for classification

[What are the main drawbacks of this method??]

Outline

- Intro: What is Statistical Machine-Learning?
- **Typology of Machine-Learning**
- General formalism for SUPERVISED Learning
- Evaluating learnt models:
  metrics for CLASSIFICATION
- Generalization *vs.* overfitting

Supervised vs Unsupervised learning

Learning is called *"supervised"* when there are "target" values for every example in training dataset:

***examples* = *(input-output)* =** $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

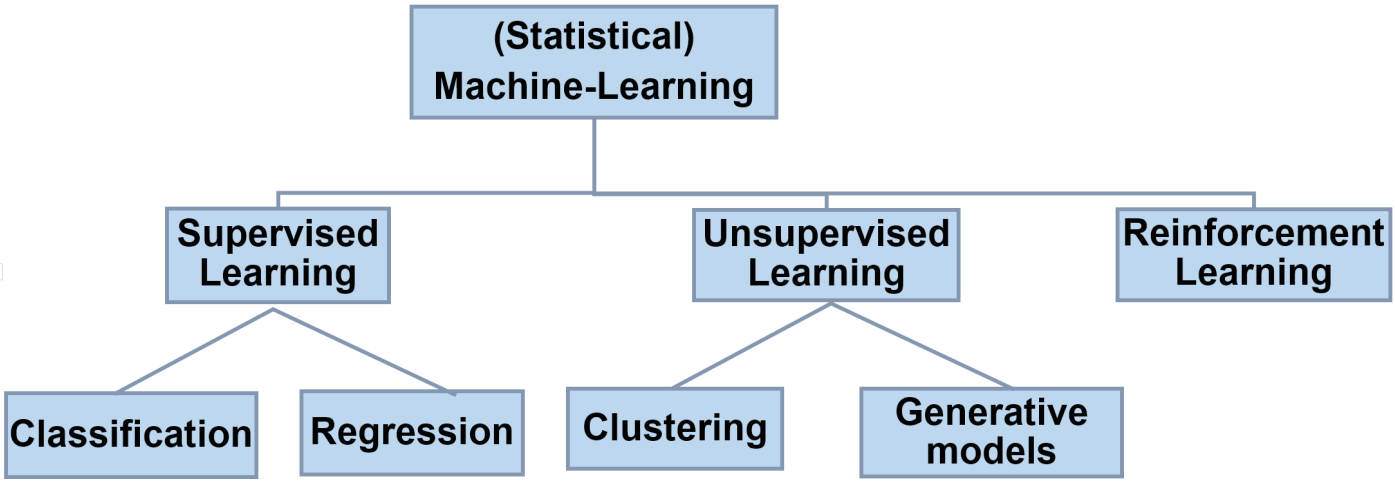
The goal is to build a (generally non-linear) approximate model for interpolation, in order to be able to GENERALIZE to input values other than those in training set

***"Unsupervised"* = when there are NO target values:**

dataset = $\{x_1, x_2, \ldots, x_n\}$

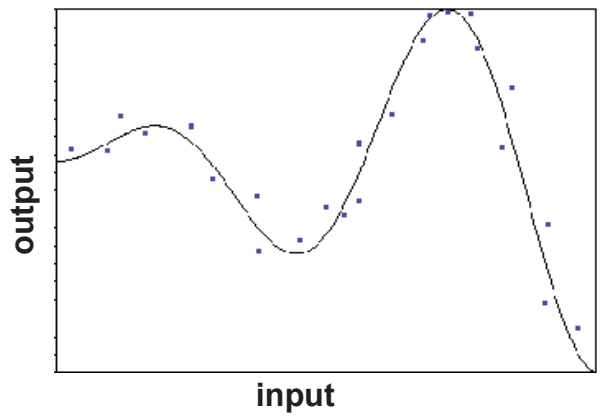
The goal is typically either to do datamining (unveil structure in the distribution of examples in input space), or to find an output maximizing a given evaluation function

Machine-Learning *TYPOLOGY*



SUPERVISED LEARNING: regression or classification

Regression

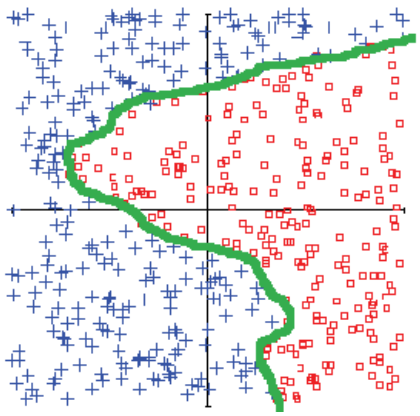


Examples $\{(x_i,y_i), i=1,\dots N\}$
x_i=input, y_i=*target output*

➔ *Infer: curve = regression* $y \approx h(x)$

y: Continuous output(s)

Classification



Input $\{x_i, i=1,\dots N\}$ *= points positions*

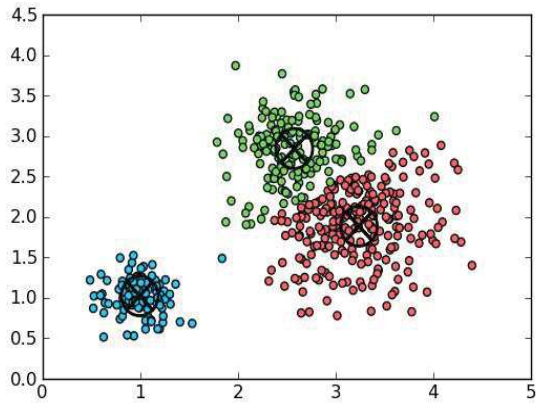
target Output = class label (□ =-1,+=+1)

➔ *Infer: label=h(x)*
(and separation boundary)

y: Discrete output(s)

UNSUPERVISED LEARNING: Clustering vs. Generative model

Clustering



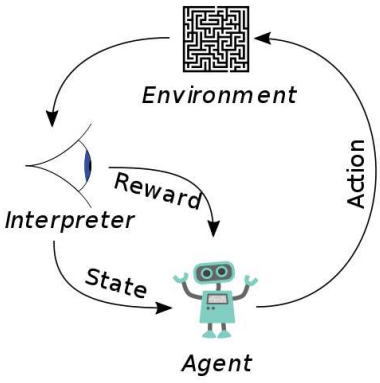
Points = examples
➔ partitioning in "groups" (colors) based on similarity

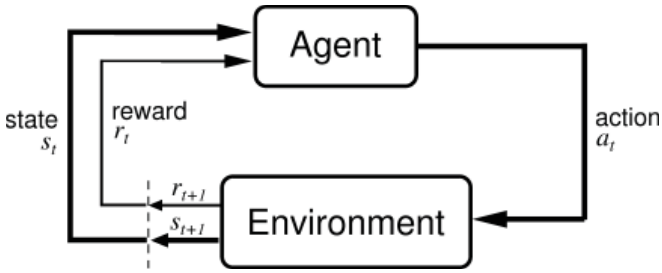
Generative model

From examples x_n, estimate the PROBABILITY DISTRIBUTION p(x)

➔ Can GENERATE new examples SIMILAR to those in training set

Reinforcement Learning (RL)





Goal: find a "policy" $a_t=\pi(s_t)$ that maximizes $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \gamma \in [0,1[$

Typical use of RL: learn a BEHAVIOR

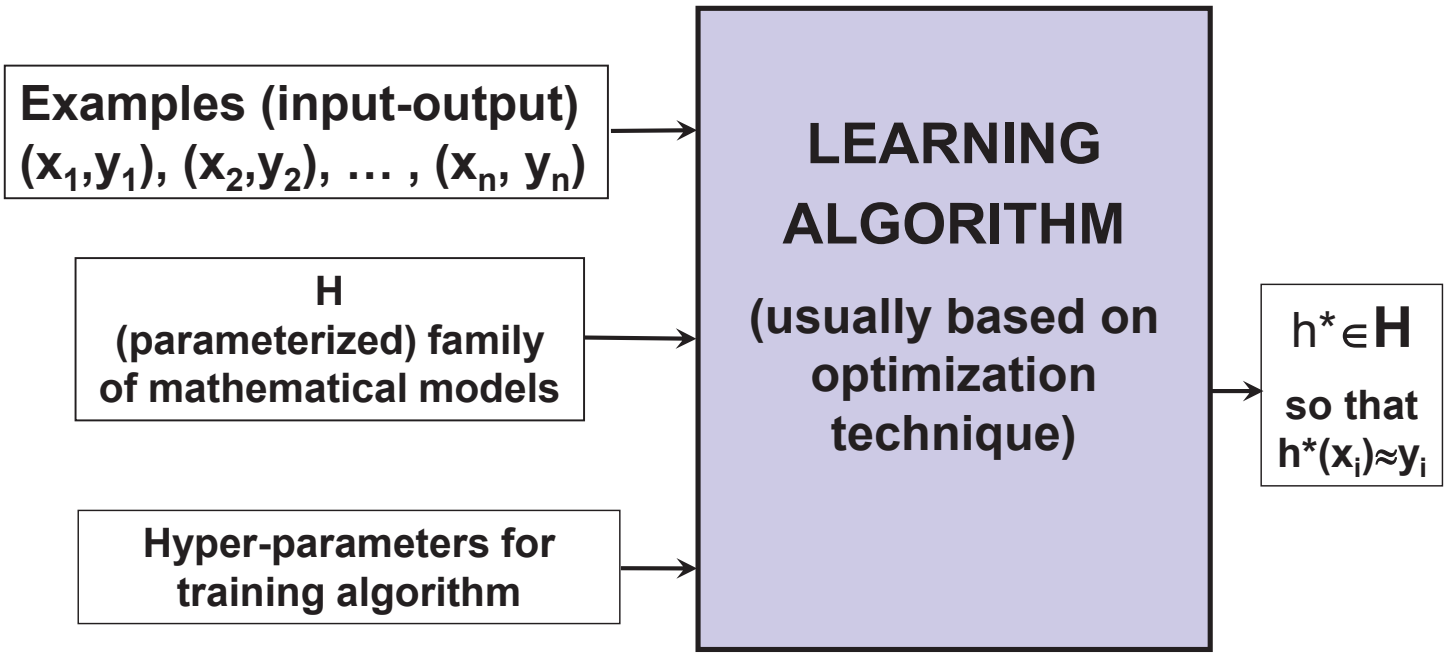
Outline

- Intro: What is Statistical Machine-Learning?
- Typology of Machine-Learning
- **General formalism for SUPERVISED Learning**
- Evaluating learnt models:
  metrics for CLASSIFICATION
- Generalization *vs.* overfitting

Many different supervised ML approaches & algorithms

- Linear regressions
- Decision trees (ID3 or CART algorithms)
- Bayesian (probabilistic) methods
- …
- Multi-layer neural networks trained with gradient backpropagation
- Support Vector Machines
- Boosting of "weak" classifiers
- Random forests
- Deep Learning (Convolutional Neural Networks,…)
- …

Supervised learning



In most cases, $h^* = \text{argMin}_{h \in H} K(h, \{(x_i,y_i)\})$ where K=cost

$K = \sum_i \text{loss}(h(x_i),y_i)$ *[+ regularization-term]* and $\text{loss}=||h(x_i)-y_i||^2$

Cost function and *loss* function

Most *supervised* Machine-Learning algorithms work by minimizing a *"cost function"*

- The cost function is generally the average over all training examples of a *"loss function"*

$$K = \sum_i \text{loss}(h(x_i),y_i)$$

(+ sometimes an additional « *regularization* » term)

- The *loss function* is usually some measure of the difference between target value and prediction by the output of the learnt model

Linear Multivariate Regression

Linear Regression, Mean Square Loss:

- decision rule: $y = W'X$
- loss function: $L(W, y^i, X^i) = \frac{1}{2}(y^i - W'X^i)^2$
- gradient of loss: $\frac{\partial L(W,y^i,X^i)}{\partial W}' = -(y^i - W(t)'X^i)X^i$
- update rule: $W(t+1) = W(t) + \eta(t)(y^i - W(t)'X^i)X^i$
- direct solution: solve linear system $[\sum_{i=1}^{P} X^i X^{i\prime}]W = \sum_{i=1}^{P} y^i X^i$

[From slide by Y. LeCun: Machine Learning and Pattern Recognition]

Logistic Multivariate Regression

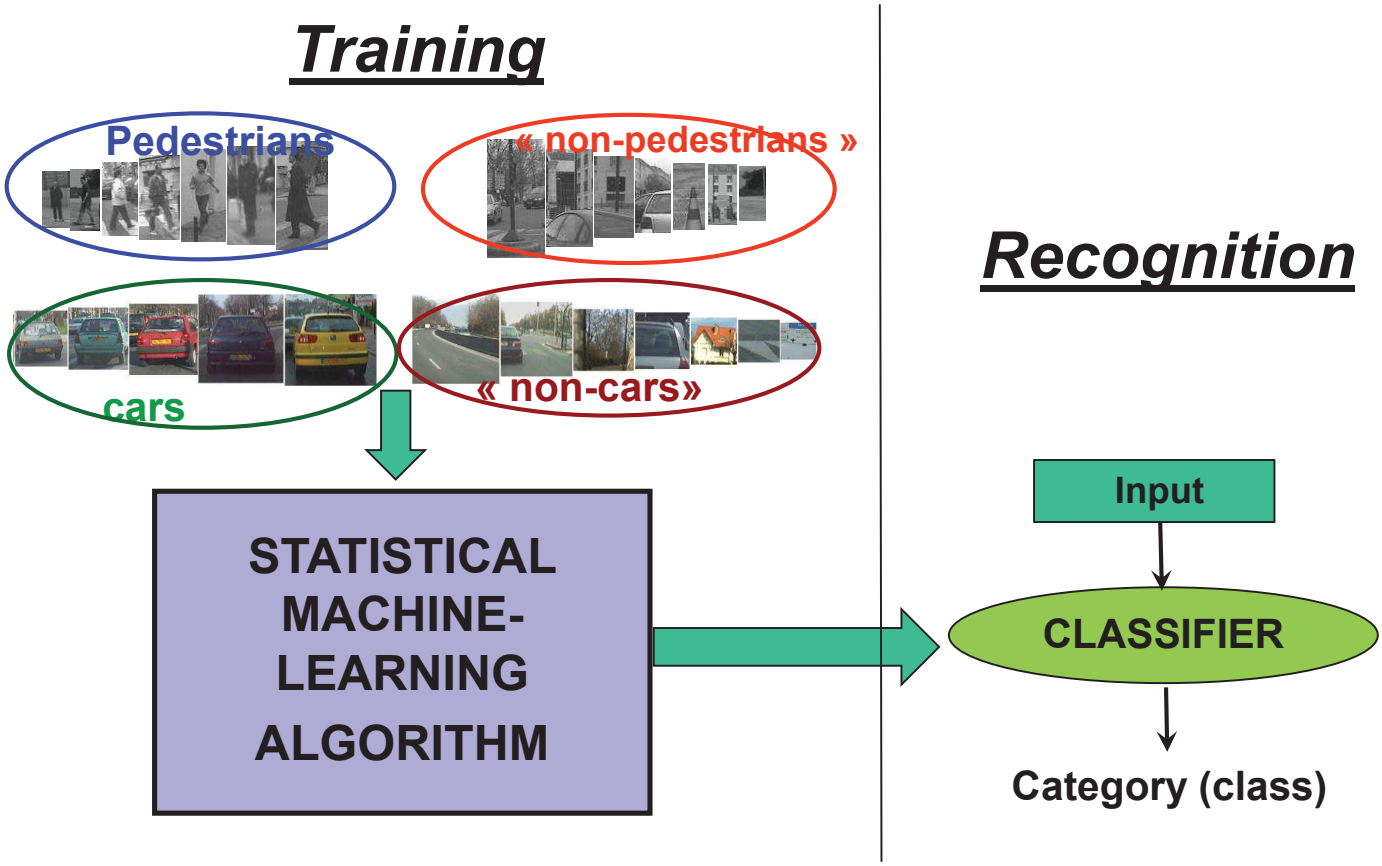
If target output is binary (classification)

Logistic Regression, Negative Log-Likelihood Loss function:

- decision rule: $y = F(W'X)$, with $F(a) = \tanh(a) = \frac{1-\exp(a)}{1+\exp(a)}$ (sigmoid function).
- loss function: $L(W, y^i, X^i) = 2\log(1 + \exp(-y^i W'X^i))$
- gradient of loss: $\frac{\partial L(W,y^i,X^i)}{\partial W}' = -\left(Y^i - F(W'X)\right)X^i$
- update rule: $W(t+1) = W(t) + \eta(t)(y^i - F(W(t)'X^i))X^i$

[From slide by Y. LeCun: Machine Learning and Pattern Recognition]

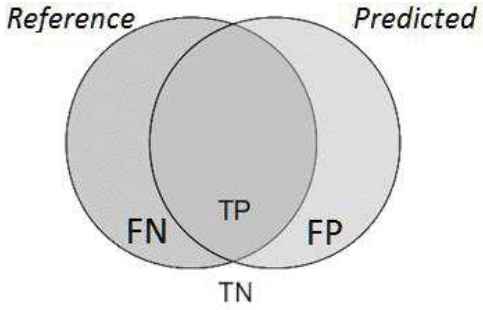
Usual two distinct phases of supervised Machine-Learning



Outline

- Intro: What is Statistical Machine-Learning?
- Typology of Machine-Learning
- General formalism for SUPERVISED Learning
- **Evaluating learnt models: metrics for CLASSIFICATION**
- Generalization *vs.* overfitting

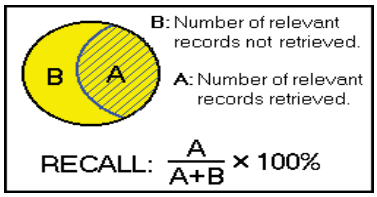
Different types of classification errors



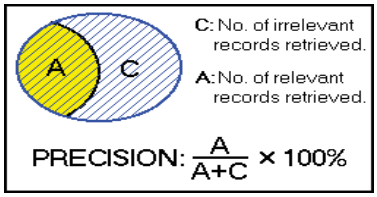
	predicted as positive	predicted as negative
positive	TP	FN
negative	FP	TN

Error rate =
(FP+FN) / (TP+TN+FP+FN)

BUT: False Negatives ("missed") ≠ False Positives!



Recall: percentage of relevant examples successfully predicted/retrieved



Precision: percentage of actually relevant examples among all those returned by the classifier

Accuracy, recall & precision formulas

	predicted as positive	predicted as negative
positive	TP	FN
negative	FP	TN

Accuracy **("correctness")** ***[en français, exactitude]*** $= \dfrac{\#\text{ of correct predictions}}{\text{Total \# of examples}} = \dfrac{TP + TN}{TP + TN + FP + FN}$

Recall **(sensitivity)** True Positive rate $= \dfrac{\#\text{ of correct } \textit{positive} \text{ predictions}}{\#\text{ of } \textit{real} \text{ positives}} = \dfrac{TP}{TP + FN}$

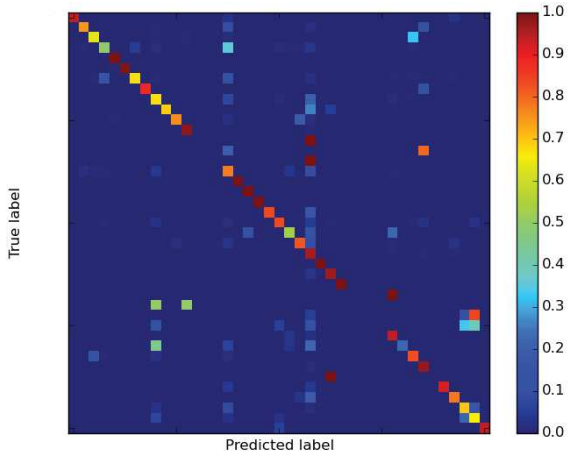
Precision **(specificity)** $= \dfrac{\#\text{ of correct } \textit{positive} \text{ predictions}}{\#\text{ of positive } \textit{predictions}} = \dfrac{TP}{TP + FP}$

Classification performance metrics

- **Accuracy** = proportion of correct
- **Recall (sensitivity)** ≈ proportion of "not missed"
  ≈ *"completeness"* level *[exhaustivité]*
- **Precision (specificity)** ≈ *reliability* of predicted labels
- **Confusion matrix**: predicted label v.s. true label

True positive	False positive
True negative	False negative

C.Matrix	1	2	3	4	5	6	ACTUAL	RECALL
1	339	15	5	0	0	0	359	94.43%
2	15	305	14	0	0	0	334	91.32%
3	6	10	242	0	0	0	258	93.80%
4	0	0	0	302	30	0	332	90.96%
5	0	0	0	15	368	0	383	96.08%
6	0	0	0	0	0	394	394	100.00%
PREDICTED	360	330	261	317	398	394	2060	94.43%
PRECISION	94.17%	92.42%	92.72%	95.27%	92.46%	100.00%	94.51%	94.66%

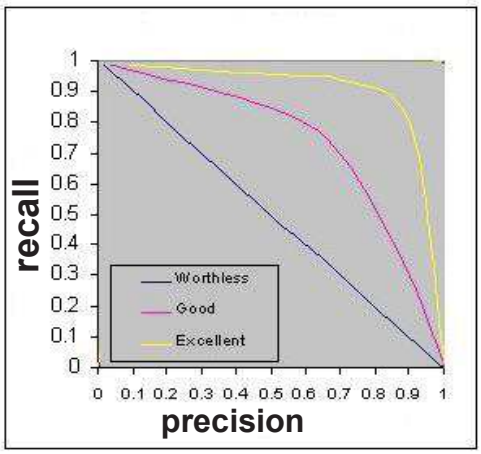


Precision-recall trade-off and curve

Classifier C1 predicts better than C2

iff C1 has better recall *and* precision

+ Trade-off between recall and precision



➔ **Compare precision-recall *curves*!**

For numeric comparison (or if curves cross each other), Area Under Curve (AUC)

Quality measures of learnt model: loss function and error types

- **Quality measure for a learnt model h:**

  $Q(h) = E(\ L(h(x),y)\ )$

  where $L(h(x),y)$ is the « ***LOSS function*** »

  generally = $||h(x)-y||^2$

- **What optimum for h?**

  h^* ***absolute*** optimum = $\text{argMin}_h(E(h))$

  h^*_H optimum ***within H family*** = $\text{argMin}_{h \in H}(E(h))$

  $h^*_{H,n}$ optimum ***in H from finite set of examples*** =

  $\text{argMin}_{h \in H}(E_n(h))$

  where $E_n(h) = (1/N)\ \Sigma_i(L(h(x_i),y_i))$

$$E(h^*_{H,n})-E(h^*) = [E(h^*_{H,n})-E(h^*_H)] + [E(h^*_H)-E(h^*)]$$

ESTIMATION error ***MODEL error***

Outline

- Intro: What is Statistical Machine-Learning?
- Typology of Machine-Learning
- General formalism for SUPERVISED Learning
- Evaluating learnt models:
  metrics for CLASSIFICATION
- **Generalization *vs.* overfitting**

Formal definition of SUPERVISED LEARNING

"LEARNING = APPROXIMATE + *GENERALIZE*"

Given a FINITE set of examples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

where $x_i \in \Re^d$ = input vectors, and $y_i \in \Re^s$ = target values

(given by the "teacher"), find a function h which

"*approximates AND GENERALIZES as best as possible*"

the underlying function such that $y_i = f(x_i) + noise$

$\Rightarrow$ goal = to minimize the GENERALIZATION error

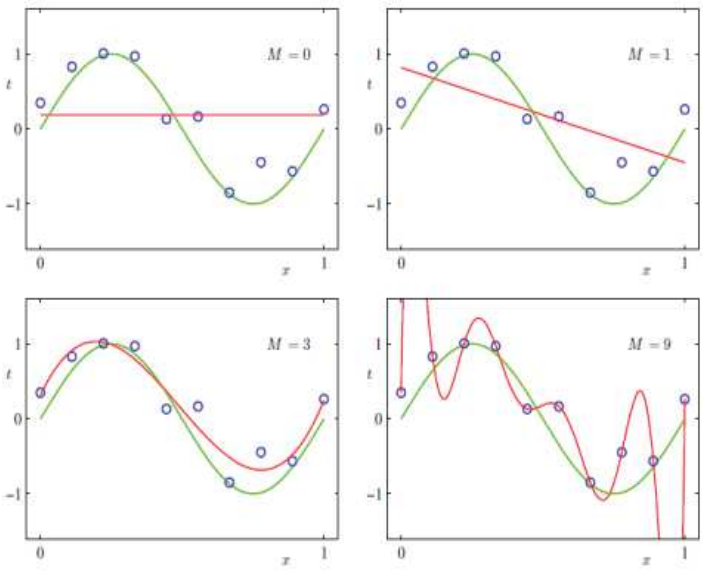
$$E_{gen} = \int \| h(x) - f(x) \|^2 \, p(x) \, dx$$

(where $p(x)$ = probability distribution of x)

About over-fitting

The generalization error cannot be directly measured, only *empirical error* on examples can be estimated:

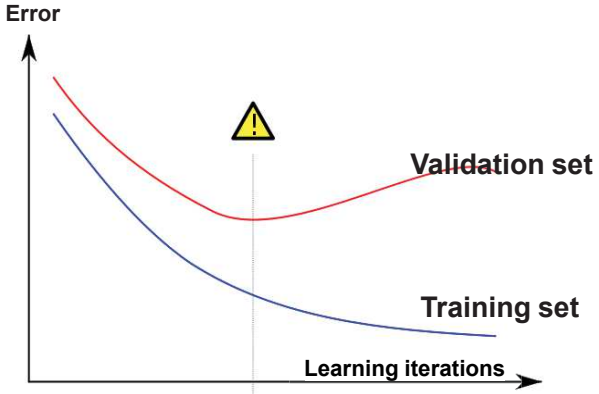
$$E_{emp} = \left(\Sigma_i \, \| h(x_i) - y_i \|^2 \right) / n$$



Fitting a data set to different orders of polynomials

[from Bishop, "Pattern Recognition and Machine Learning"]

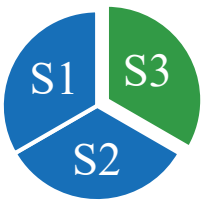
Detection of over-fitting for an iterative algorithm



Machine-Learning methodology: importance of *validation set*

To avoid over-fitting and maximize generalization, absolutely *essential to use some VALIDATION estimation,* for optimizing training hyper-parameters (and stopping criterion):

- either use a *separate validation dataset* (random split of data into Training-set + Validation-set)
- or use ***CROSS-VALIDATION***:
  - Repeat k times: train on (k-1)/k proportion of data + estimate error on remaining 1/k portion
  - Average the k error estimations



3-fold cross-validation:

- Train on S1∪S2 then estimate errS3 error on S3
- Train on S1∪S3 then estimate errS2 error on S2
- Train on S2∪S3 then estimate errS1 error on S1
- Average validation error: (errS1+errS2+errS3)/3

Empirical error and VC-dimension

- In practice, the only error that can be estimated and minimized is the *empirical* error computed on a finite set of examples:

$$E_{emp} = \left(\sum_i \|h(x_i) - y_i\|^2 \right) / n$$

- According to « regularization theory » and theoretical result by Vapnik, minimizing $E_{emp}(h)$ within heH shall also minimize E_{gen} if H has a *finite* VC-dimension

VC-dimension : *maximum* cardinal v so that for any set S of v points, all dichotomies of S can be performed by one h∈H (VC-dim ≈ complexity of H)

[VC-dimension {hyperplanes of $\mathbb{R}^n$} ?]

Regularization by adding penalty to the cost function

Vapnik has shown that:

$$\text{Proba}(\max_{h \in H} |E_{gen}(h) - E_{emp}(h)| \geq \varepsilon) < G(n, \delta, \varepsilon)$$

where n = # of examples and δ=VC-dim and G *decreases with* δ/n

$\Rightarrow$ to be sure that E_{gen} en decreases when minimizing E_{emp} , the smaller n is, the smaller the VC-dim δ needs to be

A possible way to automatically reduce VC-dim is to modify the cost function into: $C = E_{emp} + \Omega(h)$ where $\Omega(h)$ penalizes « complexity » of h ($\Rightarrow$ reduction of « effective » VC-dim)

NB: ≈ application of *"Occam's razor"* !!

(≈ "why do complicated if it can be done simpler?")

Usual form of regularization penalty: L_1 norm

In many cases, the complexity (in VC-dim sense) increases with maximum value of its parameters w_i
➔ interesting to penalize large values of w_i

Usually done by modifying cost function into
$$C = E_{emp} + \lambda \Sigma_i (||w_i||)$$

Example: LASSO = regularized linear regression
$$\text{Min}_w(\Sigma_j ||y_j - w.x_j)||_2^2 + \lambda ||w||_1)$$
[L_1-norm penalization of regressor]

NB: if using L_0 (# of NON-ZERO componants) penalization (instead of L_1), we can obtain SPARSE model

Data augmentation (for classification)

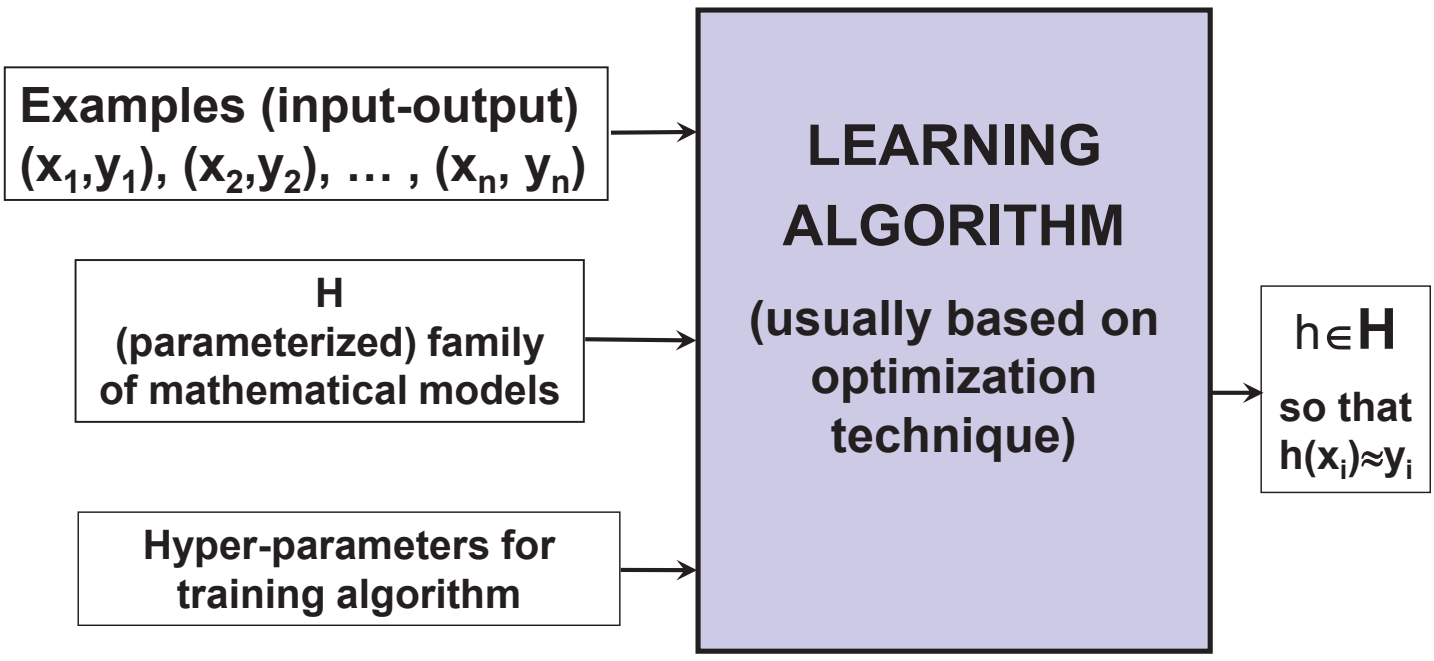
In the case of CLASSIFICATION, over-fitting avoidance and better generalization can also be favored by DATA AUGMENTATION:

- **for each labelled example in training set, generate several slightly *distorted* variants which shall have the same label**

Particularly important (and easy) for image inputs or time-series inputs

Synthesis on various algorithms for SUPERVISED Machine-Learning

Supervised learning



Summary of main shallow SUPERVISED learning algorithms

- **Decision trees:** naturally adapted to *symbolic inputs*, very fast, good scaling for very high number of classes, *"white" box*;
  BUT *noise sensitive*
- **Multi-layer neural networks:** *universal approximators*, good generalization, *easy handling of multi-class*;
  BUT optimum model NOT guaranteed, *many critical hyper-parameters* (# hidden neurons, weight init., learning rate, # training epochs,…)
- **Support Vector Machines:** *maths-guaranteed optimal separation*, possible handling of *structured input* (graphs, etc…) via kernel;
  BUT *not very efficient for multi-class* (K times 1-vs-all SVMs, or at least log(K) times Ci-vs-Cj), training computation rises quickly with input dim and # of examples O(`max(N,D) * min(N,D)^2`)
- **Boosting** of « weak » classifiers: simple algo, *can build strong classifier from any weak classifier, can select features during training*;
  BUT *not very efficient for multi-class* (n times 1-vs-all)
- **Random forests:** *OK for symbolic input*, robustness to noise, very fast to compute, *efficient for large # of classes and high input dim*;
  BUT training sometimes long

Model type choice criteria for SUPERVISED learning

	MLP Neural Network	ConvNets	SVM	Boosting	Decision Tree	Random Forest
Many classes	+	+	--	--		++
High dimension of input			-		+	++
Many examples		REQUIRED (except if transfer-learning)	-			
Interpretability (« white » box)	-	--			YES	
Data OTHER than vectors of values		Only "grid" data	Structured (string, graph)		symbolic	symbolic
Robustness to noise and erroneous labels	+	+	++		--	++
Ease/speed of training	-	---	+		++	+
Handling of features		Learn them		Automated selection		
Execution time		-			+++	+

Some REFERENCE TEXTBOOKS on Statistical Machine-Learning

- ***Introduction au machine learning***
  C. Azencott, Dunod (2018).
  https://www.dunod.com/sciences-techniques/introduction-au-machine-learning-0
- ***The Elements of Statistical Learning*** *(2nd edition)*
  T. Hastier, R. Tibshirani & J. Friedman, Springer, 2009.
  http://statweb.stanford.edu/~tibs/ElemStatLearn/
- ***Deep Learning***
  I. Goodfellow, Y. Bengio & A. Courville, MIT press, 2016.
  http://www.deeplearningbook.org/
- ***Pattern recognition and Machine-Learning***
  C. M. Bishop, Springer, 2006.
- ***Introduction to Data Mining***
  P.N. Tan, M. Steinbach & V. Kumar, AddisonWeasley, 2006.
- ***Apprentissage artificiel : concepts et algorithmes***
  A. Cornuéjols, L. Miclet & Y. Kodratoff, Eyrolles, 2002.