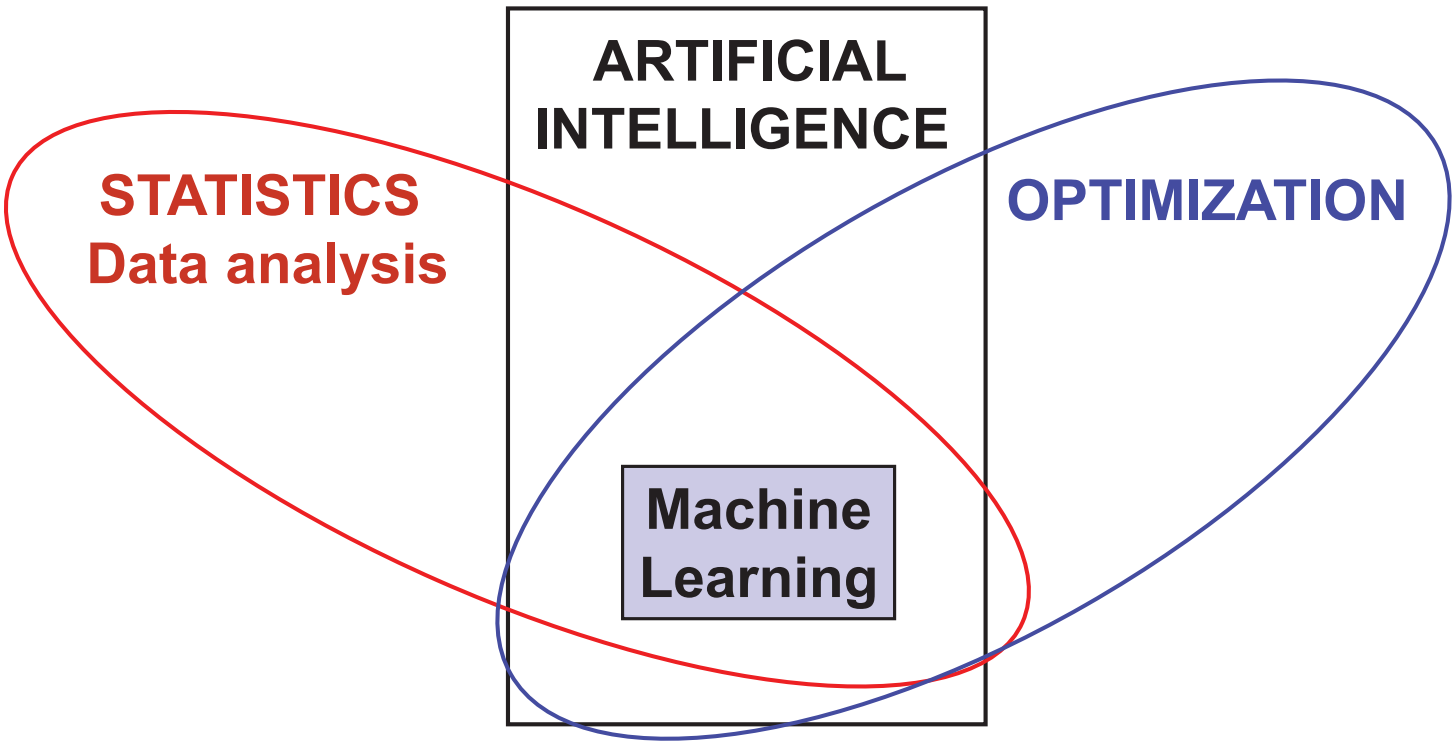
Apprentissage Artificiel
(Machine-Learning)

General framework + Supervised Learning

Pr. Fabien Moutarde
Center for Robotics
MINES ParisTech
PSL Research University

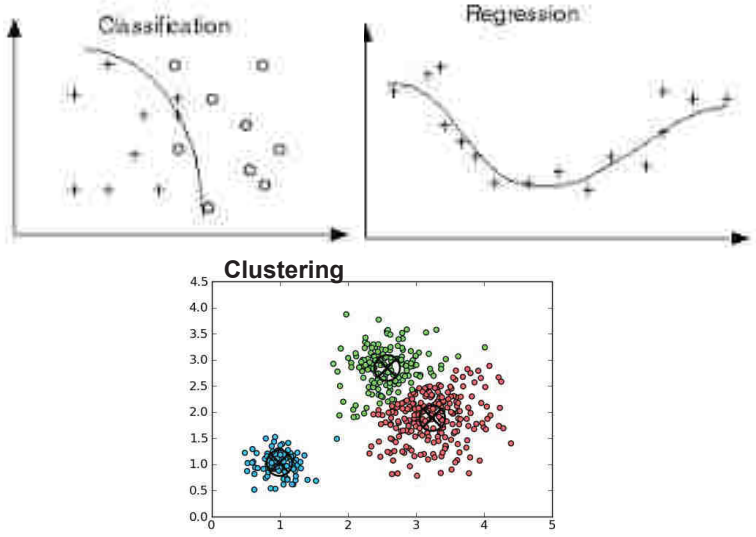
Fabien.Moutarde@mines-paristech.fr
http://people.mines-paristech.fr/fabien.moutarde

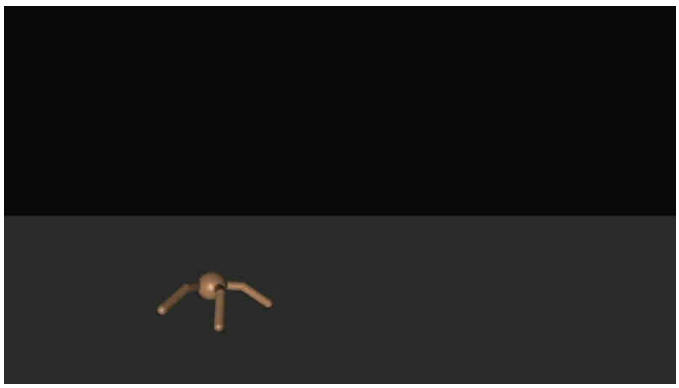
What is Statistical Machine-Learning?



What is Statistical Machine-Learning (SML)?

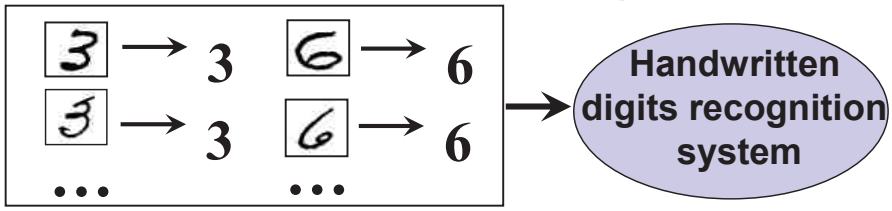
- One of many sub-fields of Artificial Intelligence
- Application of *optimization methods* to statistical modelling
- Data-driven *mathematical* modelling, for automated *classification*, *regression, partitioning/clustering*, or *decision/behavior rule*





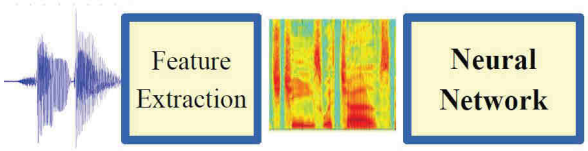
Real-world examples of SML applications

- Handwritten characters recognition



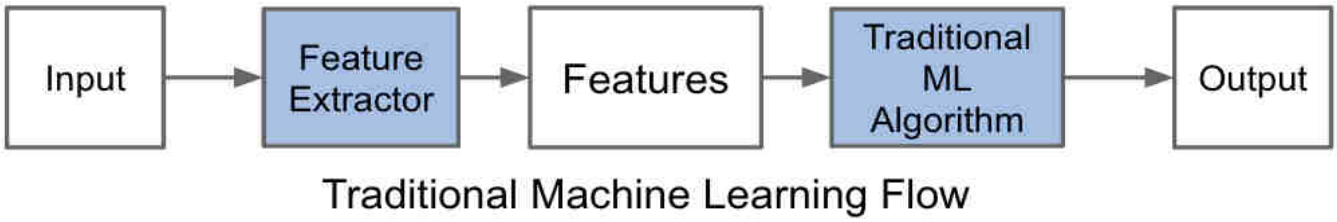
- Object category visual recognition

- Speech recognition



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

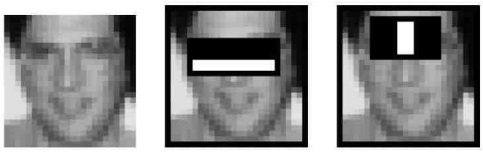
Importance of « features » in classical Machine-Learning



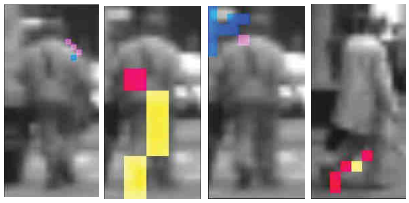
Traditional Machine Learning Flow

Examples of *hand-crafted* features

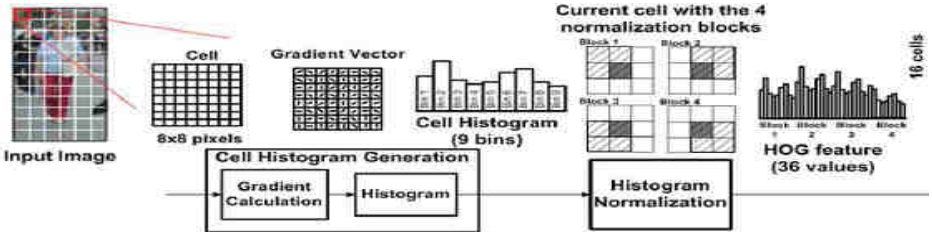
Haar features



Control-points features



HoG (Histogram of Gradients)



Machine-learning *typology*

- **What kind of (mathematical) model?**
  → polynom/spline, decision tree, neural net, kernel machine, …
- **Availability of target output data?**
  → *supervised* learning v.s. *unsupervised* learning
- **Permanent adaptability?**
  → *offline* learning v.s. *online (life-long)* learning
- **Which objective function?**
  → cost function (quadratic error, …), implicit criterium, …
- ***How* to find the best-fitting model?**
  → algorithm type (exact solving, gradient descent, quadratic optimization, heuristics, …)

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), … , (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 thos in training set

« Unsupervised » = when there are NO « target » values for examples

dataset = { x_1, x_2, … , 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/decision model that maximize an evaluation function

Cost function and *loss* function

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

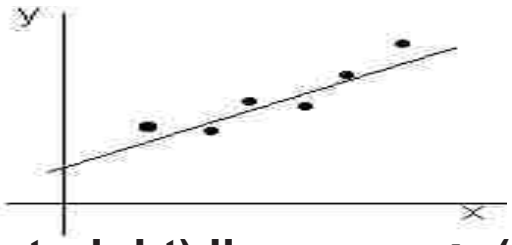
- The cost function is generally the average over all training examples of a « loss function »

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

(+ 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

Most simple Machine-Learning example: 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

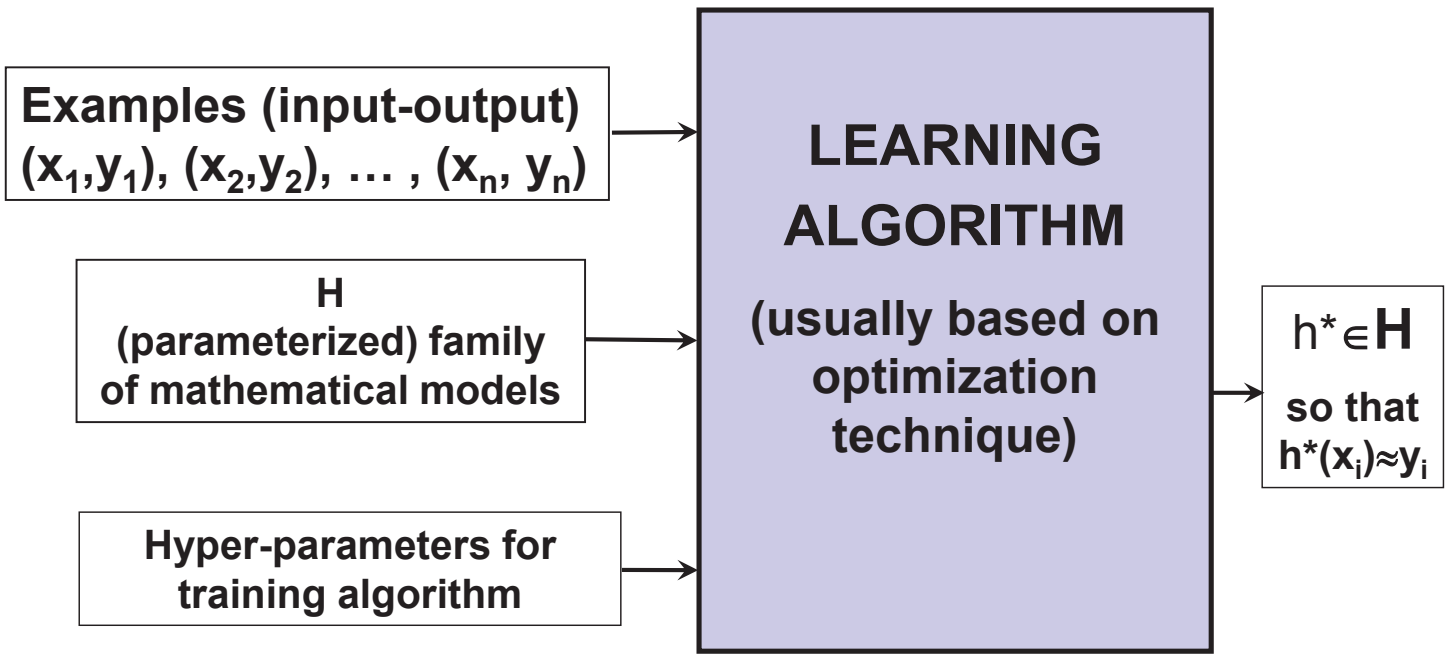
$$K=\Sigma_i (y_i-a.x_i-b)^2$$

- **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?]

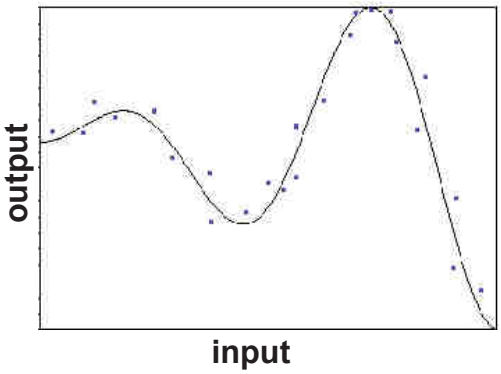
Supervised learning



In most cases, $h^*= \text{argMin}_{h\in H} K(h, \{(x_i,y_i)\})$ where K=cost
$K = \Sigma_i$ loss($h(x_i),y_i$) *[+ regularization-term]* and loss=$||h(x_i)-y_i||^2$

SUPERVISED LEARNING: regression or classification

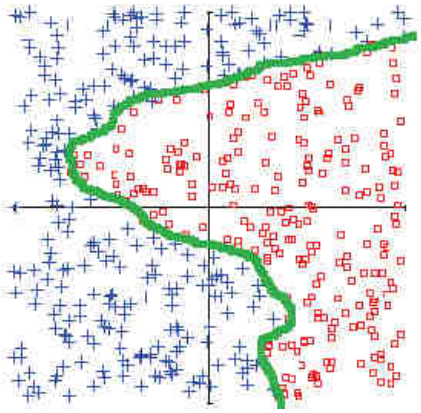
Regression



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)**

Many different 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,…)
- …

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

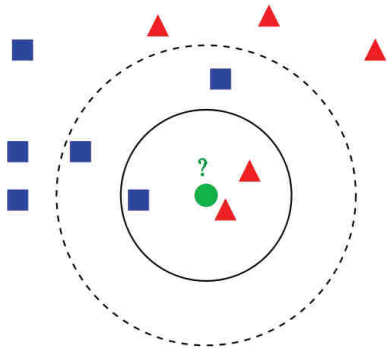
If target output is binary (classification)

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]

Nearest Neighbors algorithm



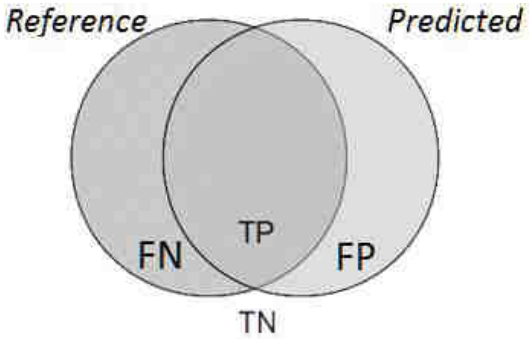
Principle of Nearest Neighbors (kNN) for classification

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

Typology of *classification* methods

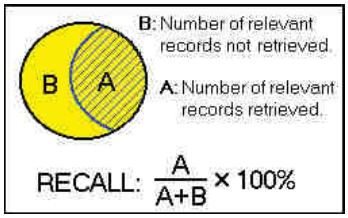
- By logistic regression
- By *similarity* ➔ Nearest Neigbors (kNN)
- By *succession of elementary tests* ➔ Decision Trees
- By *probabilistic computations* (using hypothesis on distribution of classes) ➔ Bayesian methods
- By *error minimization* (gradient descent, etc...)
  ➔ Neural Networks, etc…
  Idem + *« margin » maximization*
  ➔ Support Vector Machines (SVM)
- By *voting committee (ensemble methods)*:
  - using trees ➔ Random Forests
  - using successive weightings of examples ➔ Boosting

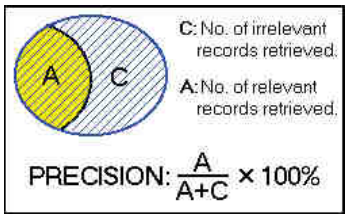
Different types of classification errors



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

False Negatives (« missed ») vs False Positives





Recall: percentage of relevant examples successfully predicted/retrieved

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

Recall and precision formulas

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

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

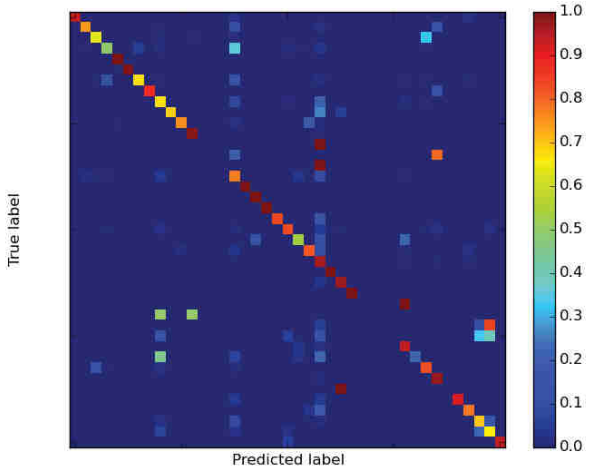
Precision (specificity) $= \dfrac{\text{Nb of correct positive predictions}}{\text{Nb of positive } \textit{predictions}} = \dfrac{TP}{TP + FP}$

Classification performance metrics

- **Recall (sensitivity)** ≈ proportion of « not missed »
  ≈ « exhaustivity » level
- **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%

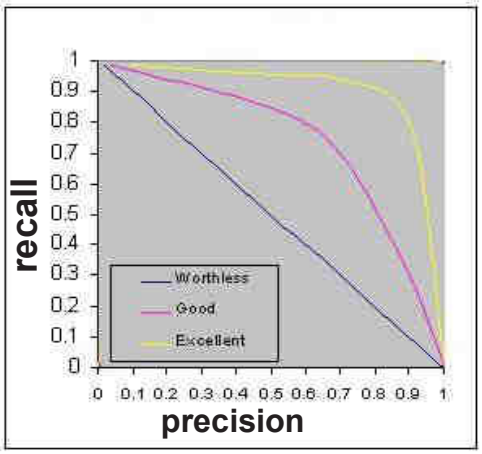


Precision-recall trade-off and curve

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)

Les diverses erreurs d'apprentissage

- **Mesure de la qualité du modèle h :**
  $E(h)=E(\ L(h(x),y)\ )$
  où $L(h(x),y)$ est la « fonction de perte »
  généralement = $||h(x)-y||^2$
- **Divers optima possibles**
  h^* optimum « absolu » = $\mathrm{argMin}_h(E(h))$
  h^*_H optimum dans H = $\mathrm{argMin}_{h\in H}(E(h))$
  $h^*_{H,n}$ optim. ds H avec ex. = $\mathrm{argMin}_{h\in H}(E_n(h))$
  où $E_n(h)=1/N\ \Sigma_i(L(h(x_i),y_i))$

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

Erreur d'ESTIMATION *Erreur de MODELE*

APPRENTISSAGE SUPERVISÉ : définition formelle

« APPRENDRE = INFERER/INDUIRE + *GENERALISER* »

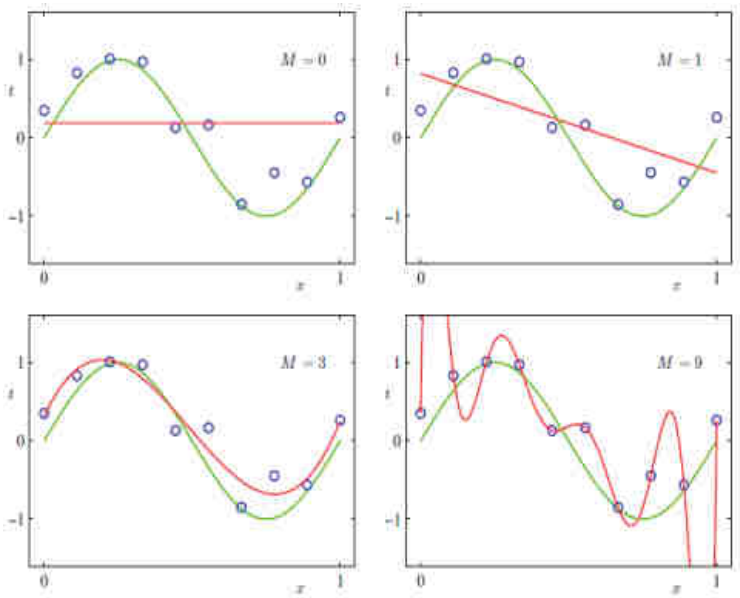
- **Etant donné un ensemble *fini* d'exemples (x_1,y_1), (x_2,y_2), … , (x_n, y_n), où $x_i\in\Re^d$ vecteurs d'entrée, et $y_i\in\Re^s$ sorties désirées (fournies par le « superviseur »), trouver une fonction h qui « *approxime et généralise au mieux* » la fonction sous-jacente f telle que $y_i=f(x_i)+$bruit**

⇒ but = minimiser erreur de généralisation

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

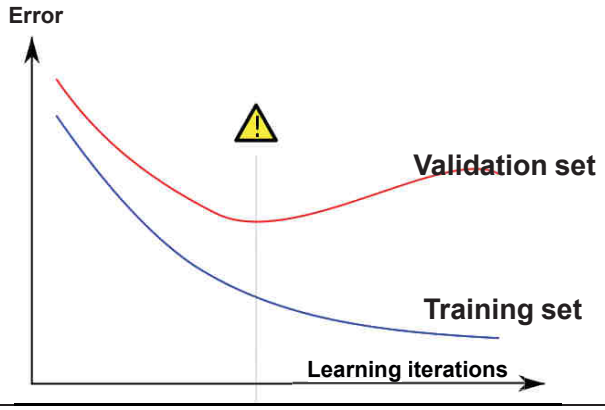
(où p(x)=distrib. de proba de x)

Notion de sur-apprentissage (overfitting)



Fitting a data set to different orders of polynomials
[from Bishop, "Pattern Recognition and Machine Learning"]

Détection de sur-apprentissage
pour un algorithme itératif



Méthodologie : notion de validation

Pour bien maximiser GENERALISATION, *importance de la validation* pour optimiser tous paramètres d'apprentissage :

- soit avec base validation séparée (découpage *aléatoire* des exemples en Apprentissage+Validation)
- soit par ***validation croisée :***
  - estimer erreur sur plusieurs ensembles de validation (k-fold ou « leave-one-out » puis en calculer la moyenne des erreurs

3-fold cross-validation :

- apprendre sur E1∪E2 et estimer sur E3
- apprendre sur E1∪E3 et estimer sur E2
- apprendre sur E2∪E3 et estimer sur E1
- moyenner errE1,errE2,errE3

Erreur empirique et VC-dimension

- En pratique, seule est mesurable l'erreur ***empirique*** sur les exemples d'apprentissage :

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

- Travaux de Vapnik et théorie de la « régularisation »

  $\Rightarrow$ minimiser $E_{emp}(h)$ sur une famille H minimisera aussi E_{gen} si H est de *VC-dimension finie*

VC-dimension : cardinal *maximum* d'un ensemble S de points, tel que pour toute dichotomie de S, il existe $h \in H$ la réalisant (en gros, la « complexité » de la famille H)

[VC-dimension {hyperplans de $\mathbb{R}^n$} ?]

Fonction de coût et terme de régularisation

- Plus précisément Vapnik a montré que :

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

  où n=nb d'ex. et δ=VC-dim, et G *décroît* si n/δ augmente

  $\Rightarrow$ pour être certain de bien minimiser E_{gen} en réduisant E_{emp}, il faut une VC-dim d'autant plus petite que n est petit

une approche possible : minimiser $C = E_{emp} + \Omega(h)$
où $\Omega(h)$ pénalise les h « trop complexes »
($\Rightarrow$ réduction de la VC-dim « effective »)

Principe similaire au « *rasoir d'Ockham* » !!
(≈ « pourquoi faire compliqué si on peut faire simple ? »)

Forme usuelle de régularisation : pénalisation L_1

Dans beaucoup de cas, la complexité du modèle augmente avec la valeur max des paramètres w_i
➔ intérêt de pénaliser les grandes valeurs de w_i

Se fait typiquement, en modifiant fonction de coût à minimiser en $C = E_{emp} + \lambda \Sigma_i (||w_i||)$

Exemple : LASSO = régression linéaire régularisée

$$Min_w(\Sigma_j ||y_j - w.x_j)||_2^2 + \lambda ||w||_1)$$

[pénalisation de la norme L_1 du régresseur]

NB : en utilisant plutôt L_0 (nb composantes NON-NULLES) pour pénalisation, on obtient modèle <u>PARCIMONIEUX (SPARSE)</u>