

Apprentissage Artificiel (Machine-Learning)

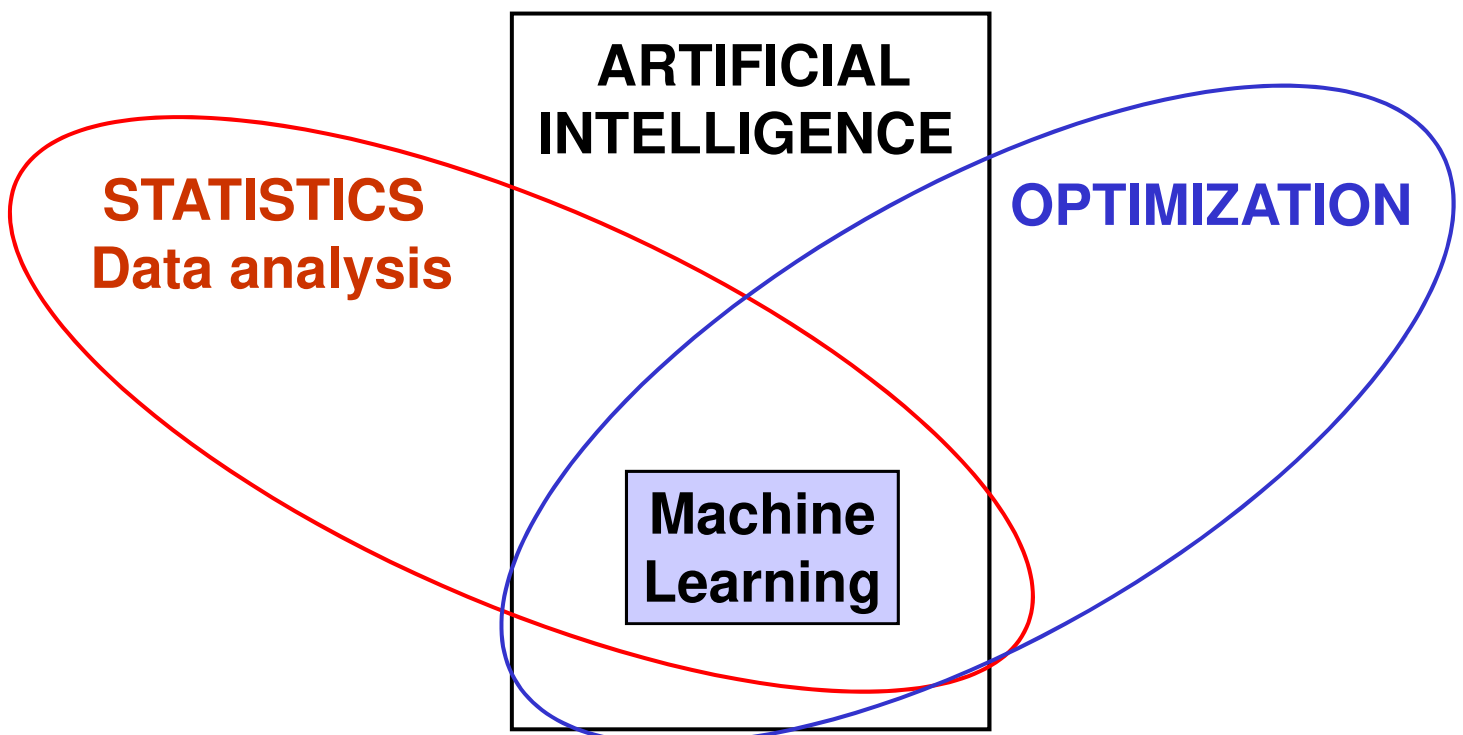
General framework + Supervised Learning

Pr. Fabien Moutarde
Robotics Lab (CAOR)
MINES ParisTech
PSL Research University

Fabien.Moutarde@mines-paristech.fr

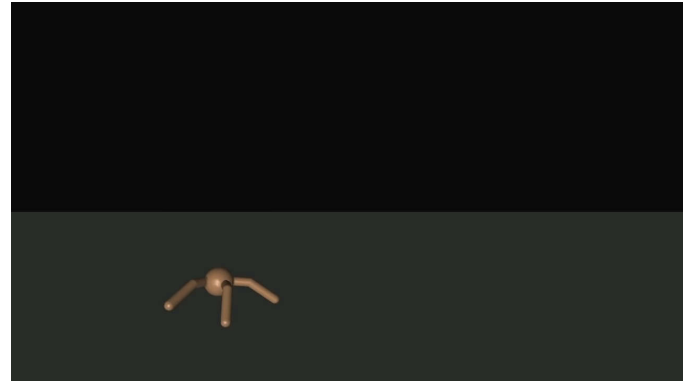
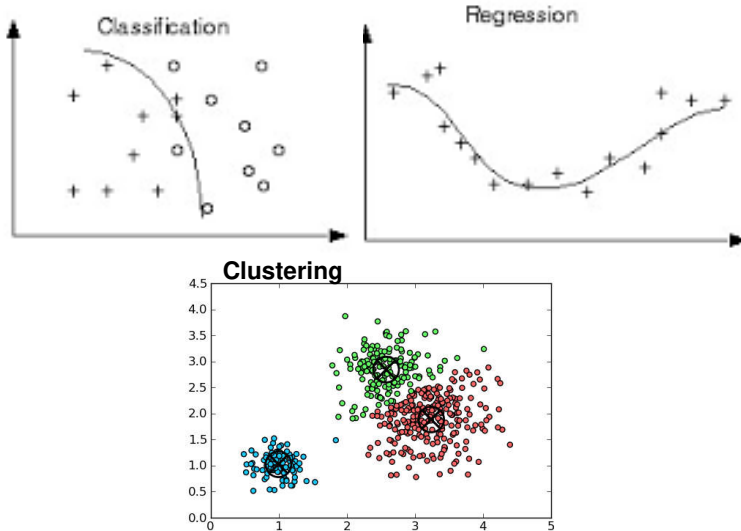
<http://people.mines-paristech.fr/fabien.moutarde>

What is Statistical Machine-Learning?



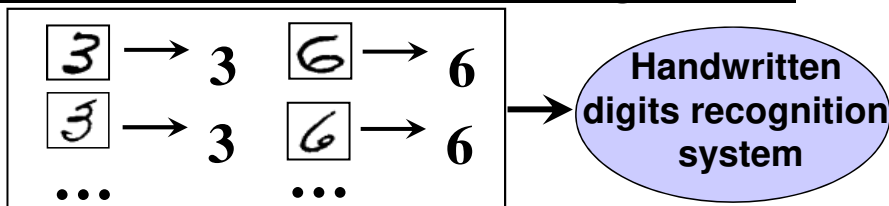
What is Statistical Machine-Learning (SML)?

- 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

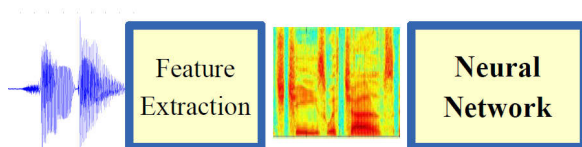
- Handwritten characters recognition



- Object category visual recognition



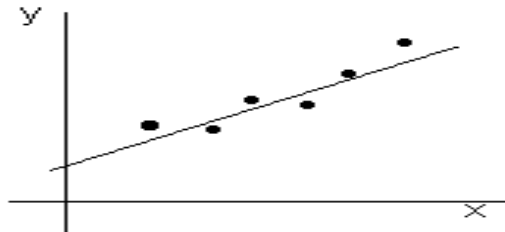
- Speech recognition



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

- **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, ...)

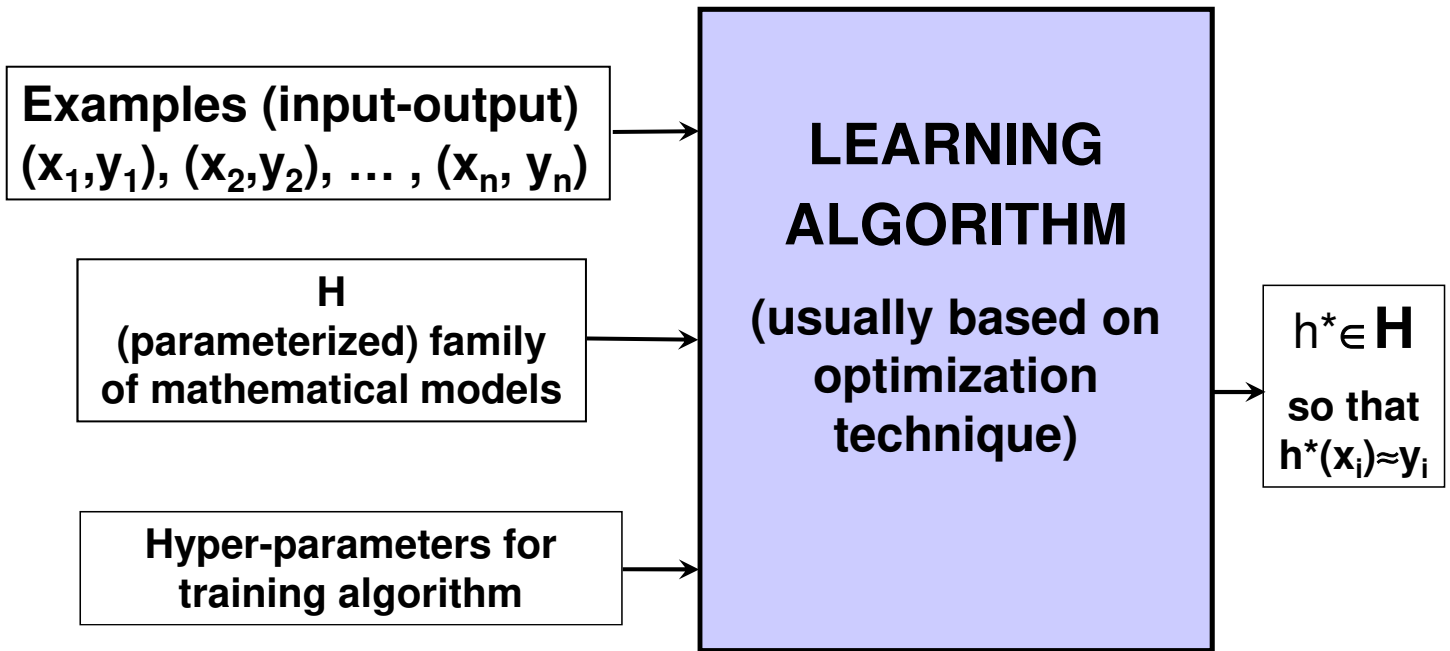
Most simple Machine-Learning example: Least Squares Linear Regression



- **Model:** (straight) line $y=ax+b$ (2 parameters a and b)
- **Data:** n points with target value $(x_i, y_i) \in \mathbb{R}^2$
- **Cost function:** sum of squares of deviation from line
$$K = \sum_i (y_i - a \cdot x_i - b)^2$$
- **Algorithm:** direct (or iterative) solving of linear system

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

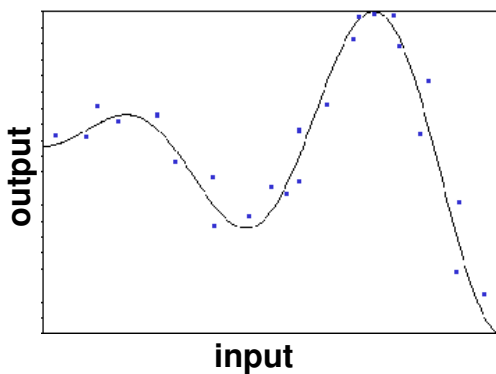
[Question: Where does this equation come from?]



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

SUPERVISED LEARNING: regression or classification

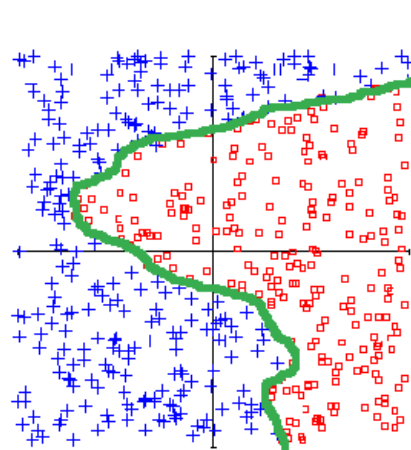
Regression



points = examples \rightarrow curve = regression

Continuous output(s)

Classification



Input =
point position
target Output =
class label
($\square = -1, + = +1$)
 \downarrow
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 $[\sum_{i=1}^P X^i X^{i'}]W = \sum_{i=1}^P y^i X^i$

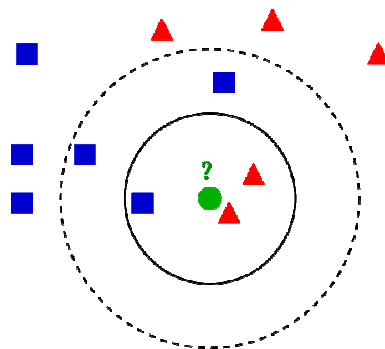
If target output is binary (classification)

Logistic Regression, Negative Log-Likelihood Loss function:

- decision rule: $y = F(W'X)$, with $F(a) = \frac{1 - \exp(a)}{1 + \exp(a)}$ (sigmoid function).
- loss function: $L(W, y^i, X^i) = 2 \log(1 + \exp(-y^i W'X^i))$
- gradient of loss: $\frac{\partial L(W, y^i, X^i)}{\partial W} = -(Y^i - F(W'X)) X^i$
- 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

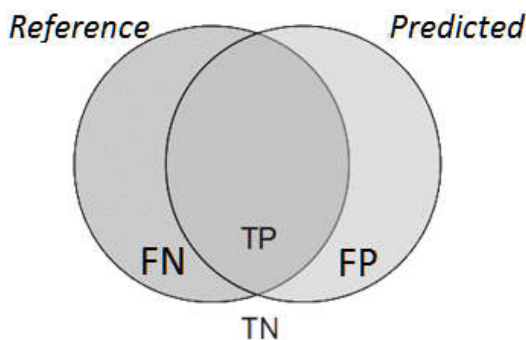


Principle of Nearest Neighbors (kNN) for classification

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

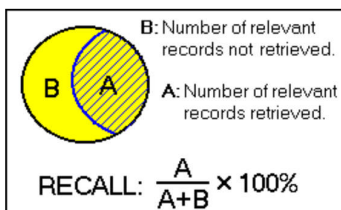
- By logistic regression
- By *similarity* → Nearest Neighbors (kNN)
- By *succession of elementary tests* → Decision Trees
- By *probabilistic computations* (using hypothesis on distribution of classes) → Bayesian methods
- By *error minimization* (gradient descent, etc...)
 - Neural Networks, etc...
- 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

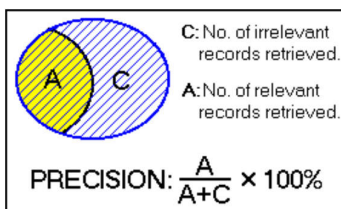


| | | |
|----------|-----------------------|-----------------------|
| | predicted as positive | predicted as negative |
| positive | TP | FN |
| negative | FP | TN |

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 |

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

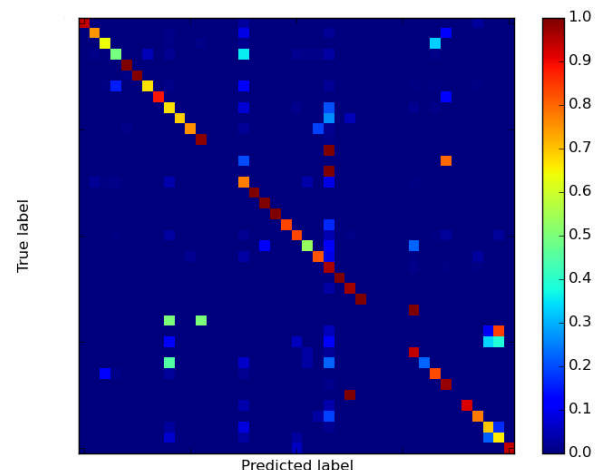
$$\text{Precision (specificity)} = \frac{\text{Nb of correct positive predictions}}{\text{Nb of positive *predictions*}} = \frac{TP}{TP + FP}$$

Classification performance metrics

- **Recall (sensitivity)** \approx proportion of « not missed » \approx « exhaustivity » level
- **Precision (specificity)** \approx reliability of predicted labels
- **Confusion matrix**: predicted label v.s. true label

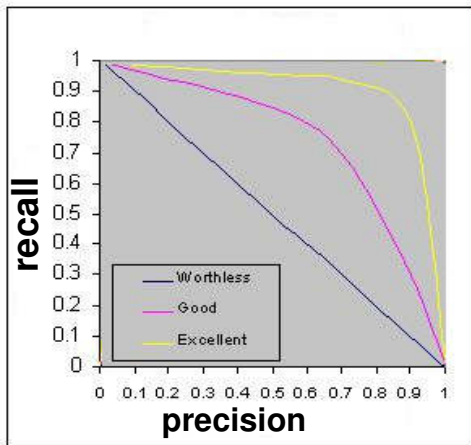
| | |
|---------------|----------------|
| True positive | False positive |
| True negative | False negative |

| 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% |



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 :**

$$\mathbf{E}(\mathbf{h}) = \mathbf{E}(\mathbf{L}(\mathbf{h}(\mathbf{x}), \mathbf{y}))$$

où $\mathbf{L}(\mathbf{h}(\mathbf{x}), \mathbf{y})$ est la « fonction de perte »

$$\text{généralement} = \|\mathbf{h}(\mathbf{x}) - \mathbf{y}\|^2$$

- **Divers optima possibles**

$$\mathbf{h}^* \text{ optimum « absolu »} = \text{argMin}_{\mathbf{h}}(\mathbf{E}(\mathbf{h}))$$

$$\mathbf{h}^*_{\mathbf{H}} \text{ optimum dans } \mathbf{H} = \text{argMin}_{\mathbf{h} \in \mathbf{H}}(\mathbf{E}(\mathbf{h}))$$

$$\mathbf{h}^*_{\mathbf{H}, \mathbf{n}} \text{ optim. ds } \mathbf{H} \text{ avec ex.} = \text{argMin}_{\mathbf{h} \in \mathbf{H}}(\mathbf{E}_{\mathbf{n}}(\mathbf{h}))$$

$$\text{où } \mathbf{E}_{\mathbf{n}}(\mathbf{h}) = 1/\mathbf{N} \sum_i (\mathbf{L}(\mathbf{h}(\mathbf{x}_i), \mathbf{y}_i))$$

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

Erreur d'ESTIMATION

Erreur de MODELE

« **APPRENDRE = INFÉRER/INDUIRE + GENERALISER** »

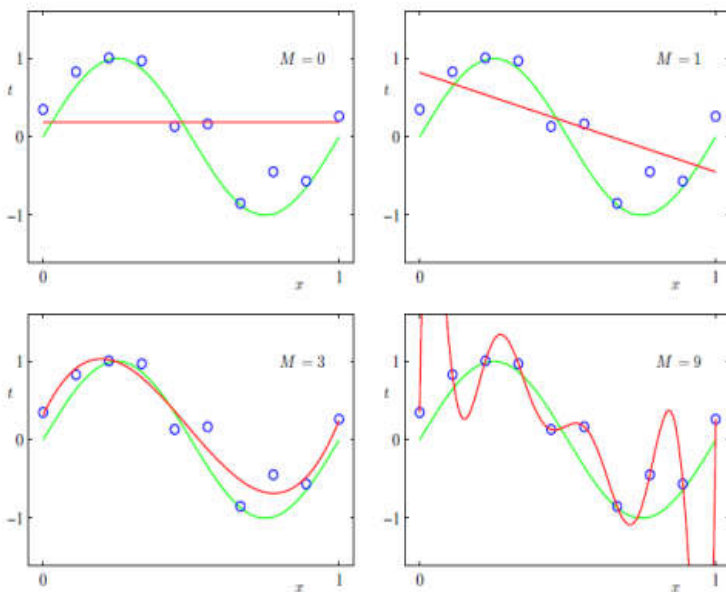
- Etant donné un ensemble *fini* d'exemples $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, où $x_i \in \mathcal{R}^d$ vecteurs d'entrée, et $y_i \in \mathcal{R}^s$ sorties désirées (fournies par le « superviseur »), trouver une fonction h qui « approxime et généralise au mieux » la fonction sous-jacente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

⇒ but = minimiser erreur de généralisation

$$E_{\text{gen}} = \int \|h(\mathbf{x}) - f(\mathbf{x})\|^2 p(\mathbf{x}) d\mathbf{x}$$

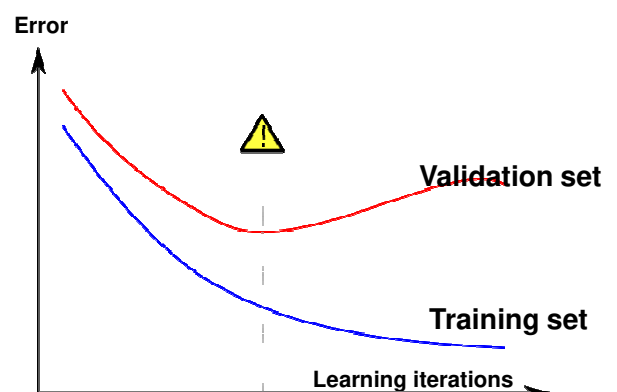
(où $p(\mathbf{x}) = \text{distrib. de proba de } \mathbf{x}$)

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



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 $E1 \cup E2$ et estimer sur $E3$
- apprendre sur $E1 \cup E3$ et estimer sur $E2$
- apprendre sur $E2 \cup E3$ 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 :

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

- Travaux de Vapnik et théorie de la « régularisation »
 \Rightarrow minimiser $E_{\text{emp}}(h)$ 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 :

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

où $n = \text{nb d'ex.}$ et $\delta = \text{VC-dim}$, et G décroît si n/δ augmente

⇒ 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 $C = E_{\text{emp}} + \Omega(h)$

où $\Omega(h)$ pénalise les h « trop complexes »

(⇒ 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 $C = E_{\text{emp}} + \lambda \sum_i (\|w_i\|)$

Exemple : LASSO = régression linéaire régularisée

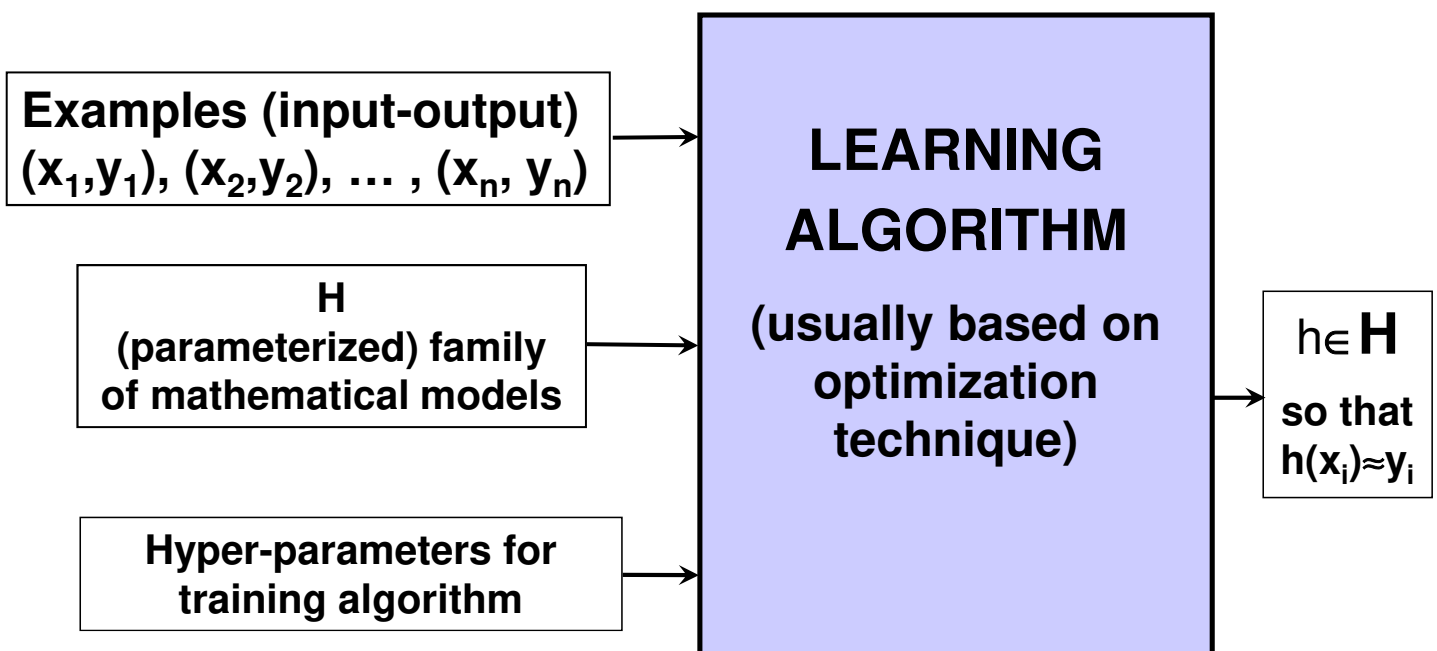
$$\text{Min}_w (\sum_j \|y_j - w \cdot x_j\|_2^2 + \lambda \|w\|_1)$$

[pénalisation de la norme L_1 du régresseur]

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

- 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 learning



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

- **Decision trees**: naturally adapted to symbolic input, very fast, « white » box
- **Multi-layer neural networks**:
- **Support Vector Machines**:
- **Boosting** of « weak » classifiers
- **Random forests**:
- **Deep Learning** (Convolutional Neural Networks,...)

Critères de choix d'algo pour apprentissage *SUPERVISÉ*

| | Réseau neurones à couches (MLP) | Conv. Net (Deep Learning) | SVM | Boosting | Arbres de décision | Forêt aléatoire |
|--|---------------------------------|---------------------------|-------------------------------|-----------|--------------------|-----------------|
| Robustesse à données bruitées / aberrantes | + | ++ | ++ | | -- | ++ |
| Temps/facilité d'apprentissage | - | --- | + | | ++ | |
| Temps de calcul | | -- | | | +++ | + |
| Beaucoup de classes | + | | -- | -- | | ++ |
| Grande dimension entrées | | | - | | + | ++ |
| Beaucoup d'exemples | | REQUIS | - | | | |
| Compréhensibilité (boite « blanche ») | | | | | OUI | |
| Données AUTRES que vecteurs numériques | | | Structurées (string, graphes) | | symbolique | symbolique |
| Gestion des features | | Apprentissage | | Sélection | | |

- ***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, AddisonWesley, 2006.
- ***Machine Learning***
T. Mitchell, McGraw-Hill Science/Engineering/Math, 1997.
- ***Apprentissage artificiel : concepts et algorithmes***
A. Cornuéjols, L. Miclet & Y. Kodratoff, Eyrolles, 2002.