Apprentissage Artificiel (Statistical Machine-Learning)

General framework + Supervised Learning

Assoc. Prof. Sascha Hornauer

Center for Robotics
Mines Paris
PSL Université

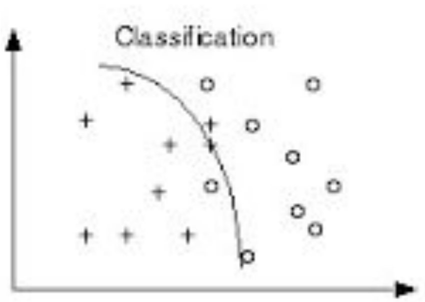
sascha.hornauer@minesparis.psl.eu

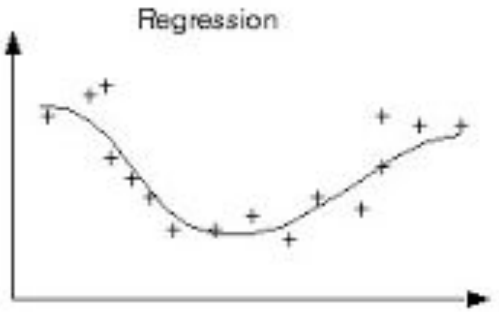
Outline

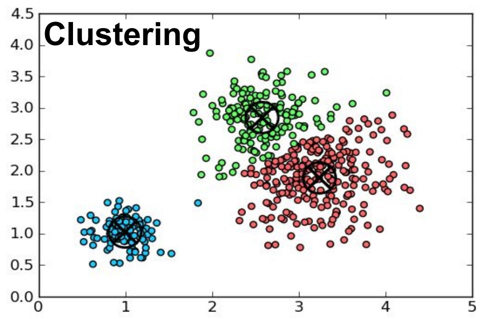
- Intro: What is Statistical Machine-Learning?
- Typology of Machine-Learning
- General formalism for SUPERVISED Learning
- Evaluating learnt models:
  metrics for CLASSIFICATION
- Generalization *vs.* overfitting

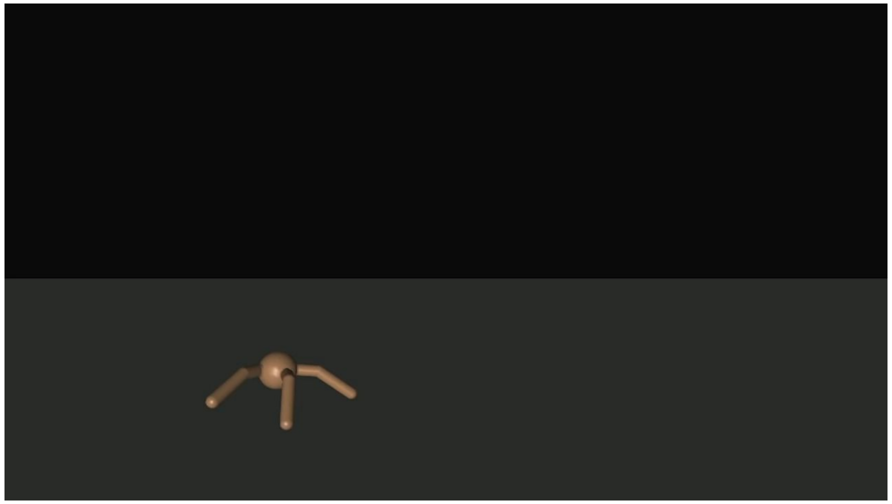
Statistical Machine-Learning

- 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*

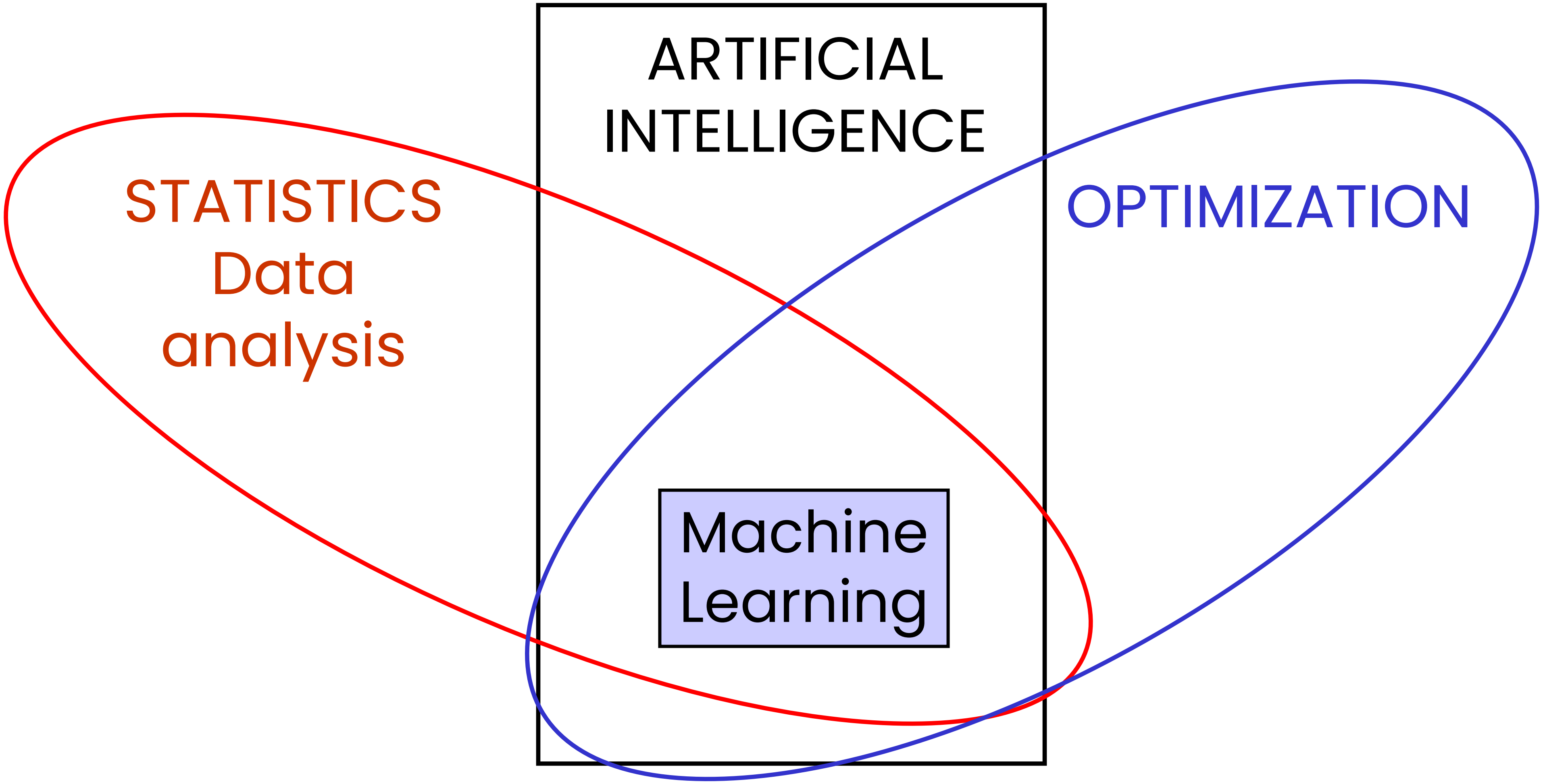






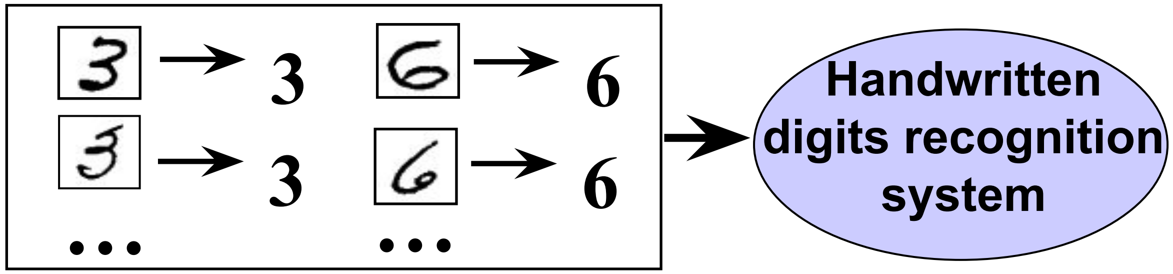


What is Statistical Machine-Learning?

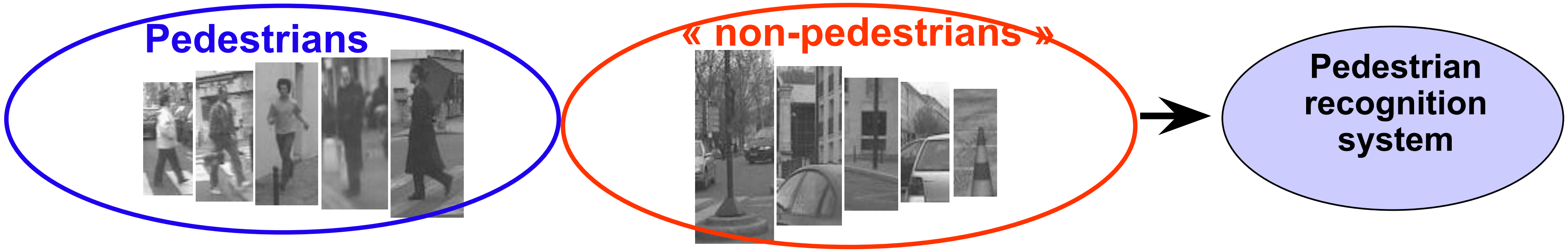


Real-world examples of Machine-Learning applications

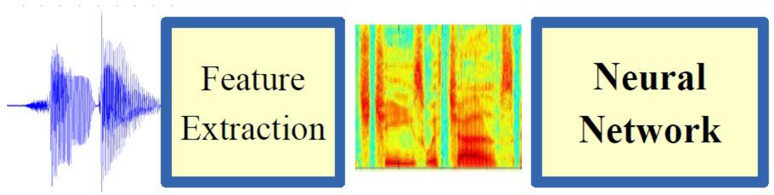
- **Handwritten characters recognition**



- **Object category visual recognition**

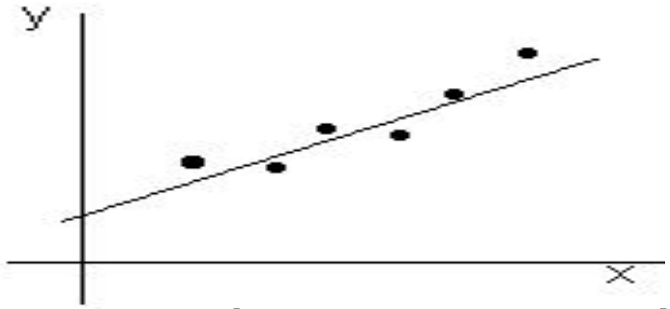


- **Speech recognition**



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

One of simplest ML algorithm: Least Squares Linear Regression



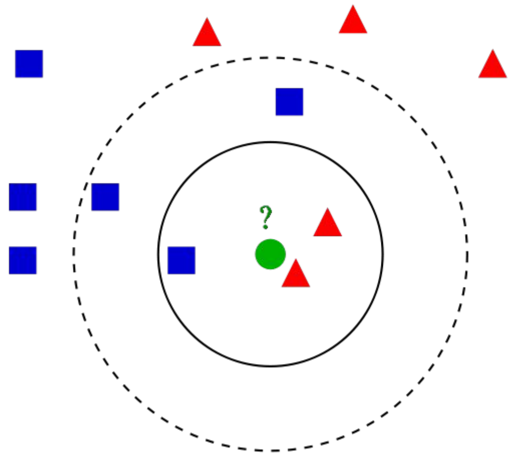
- **Model:** (straight) line `y=ax+b` (2 parameters `a` and `b`)
- **Data:** n points with target value $(x_i,y_i) \in \Re^2$
- **Cost function:** sum of squares of deviation from line

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

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

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

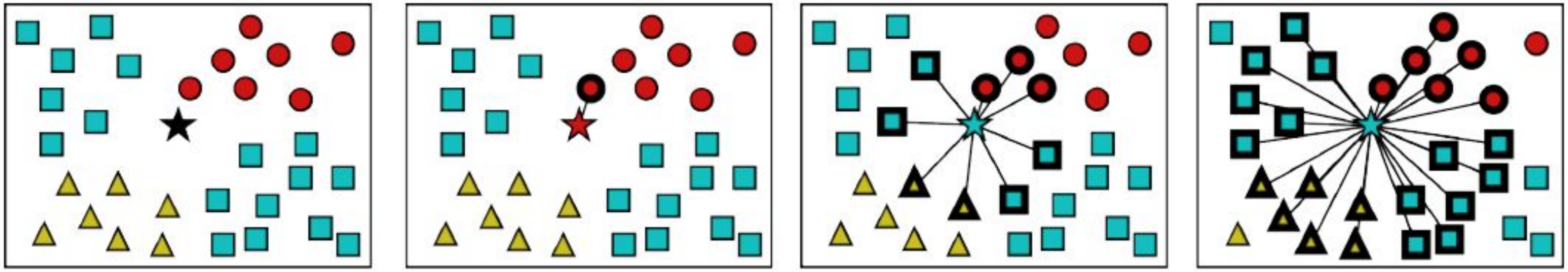
Simplest *classification* method: Nearest Neighbors algorithm



Principle of Nearest Neighbors (kNN) for classification

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

k-Nearest Neighbors



- Some outlier vectors get ‘outvoted’ with high enough number **k** of neighbors

Cmp. Chapter 5.1.1. Computer Vision: Algorithms and Applications 2nd Edition, Szeliski 2021

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 *"<u>supervised</u>"* when <u>there are "target" values</u> for every example in training dataset:

examples = (input , output) = $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_n, \mathbf{y}_n)$

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

***"<u>Unsupervised</u>"* = when there are <u>NO target values</u>:**

dataset = $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$

The goal is typically either to do datamining (unveil structure in the distribution of examples in input space)

Types of Machine-learning

- **Availability of target output data?**
  → ***Supervised*** **learning vs.** ***Unsupervised*** **learning or** ***Reinforcement*** **Learning**

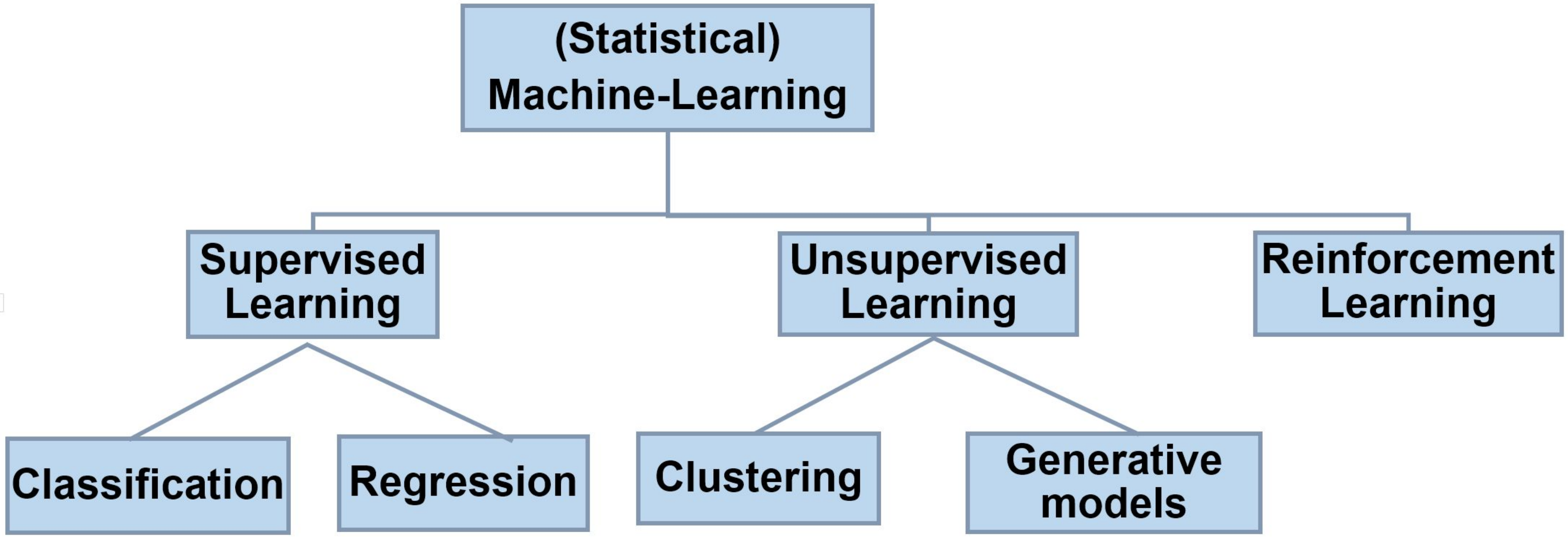
- **Permanent adaptability?**
  → ***offline*** **learning vs.** ***online (life-long)*** **learning**

- **What kind of (mathematical) model?**
  → **polynom/spline, decision tree, neural net, kernel machine, …**

- **Which objective function?**
  → **cost function (quadratic error, …), implicit criterium, …**

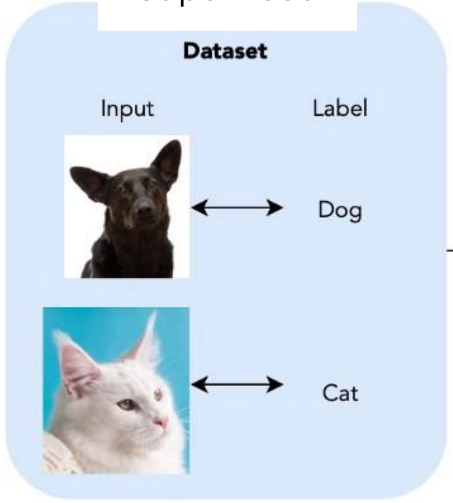
- ***How*** **to find the best-fitting model?**
  → **algorithm type (exact solving, gradient descent, quadratic optimization, heuristics, …)**

Machine-Learning *TYPOLOGY*

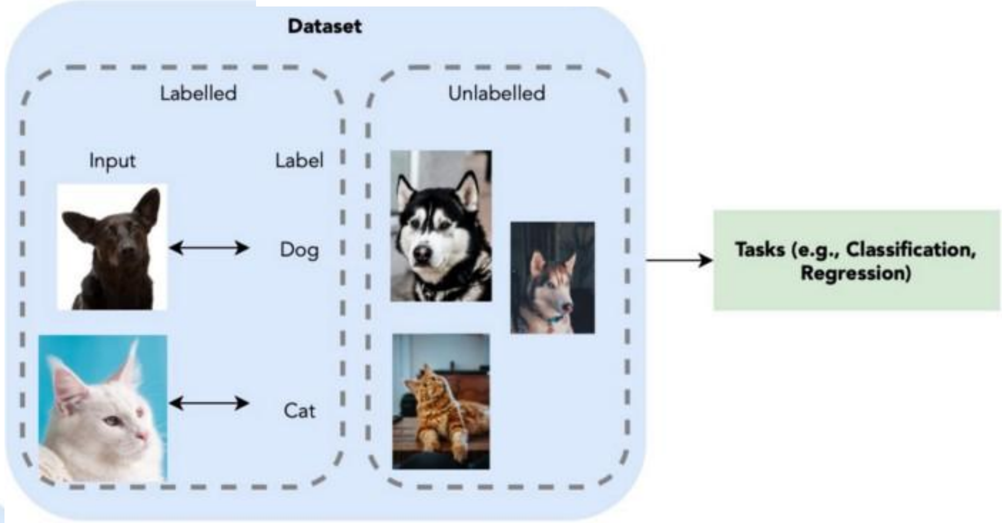


Un- or Self-Supervised Training?

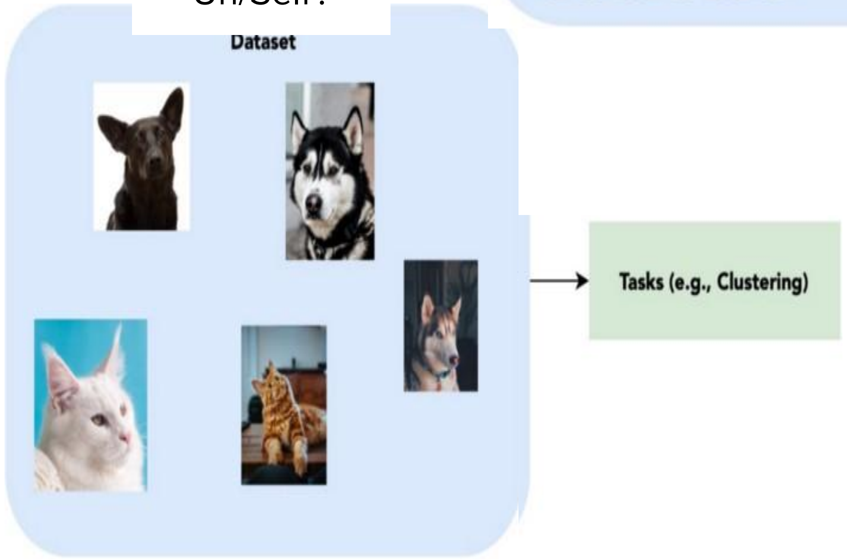
Supervised



Semi-Supervised



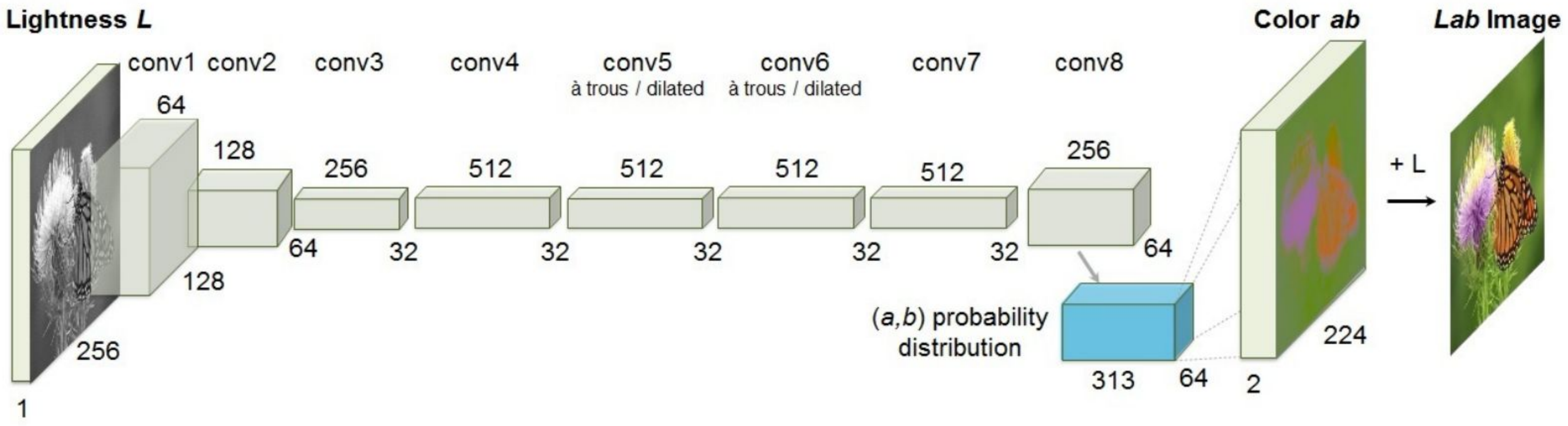
Un/Self?



Use statistical properties of data itself without human labels.

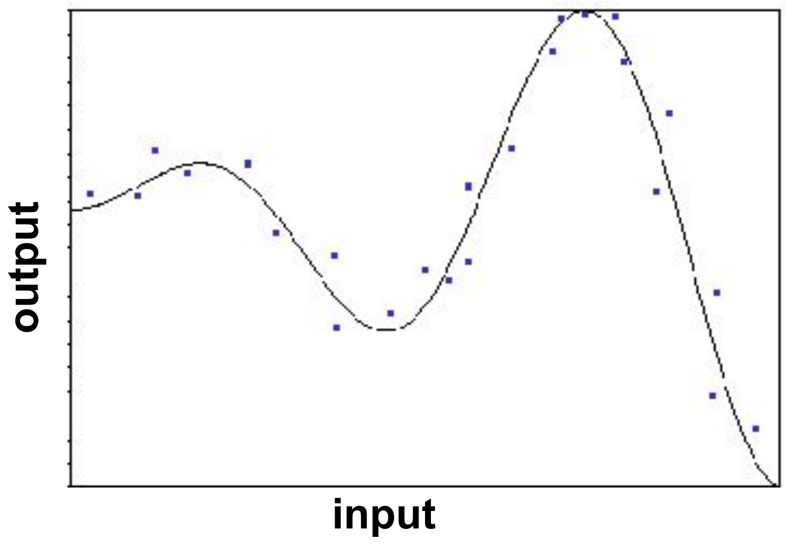
Images: https://towardsdatascience.com/supervised-semi-supervised-unsupervised-and-self-supervised-learning-7fa79aa9247c

Self-Supervised Learning



SUPERVISED LEARNING: regression or classification

Regression

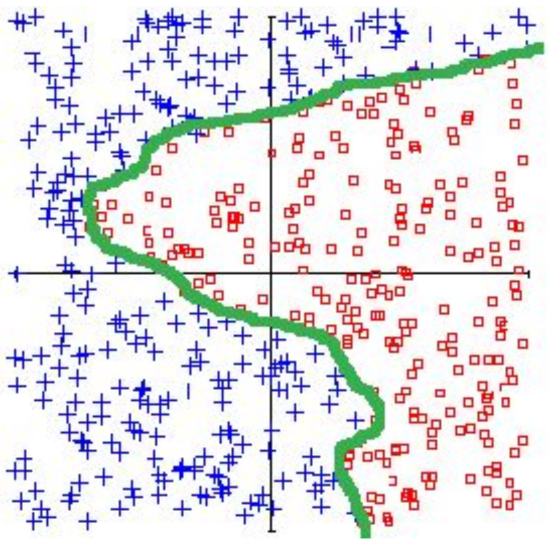


***Examples* {(x_i,y_i), i=1,…N}**
x_i=input, y_i=*target output*

Infer: curve = regression y ≈ h(x)

y: <u>Continuous</u> output(s)

Classification



Input* {x_i, i=1,…N} *= points positions

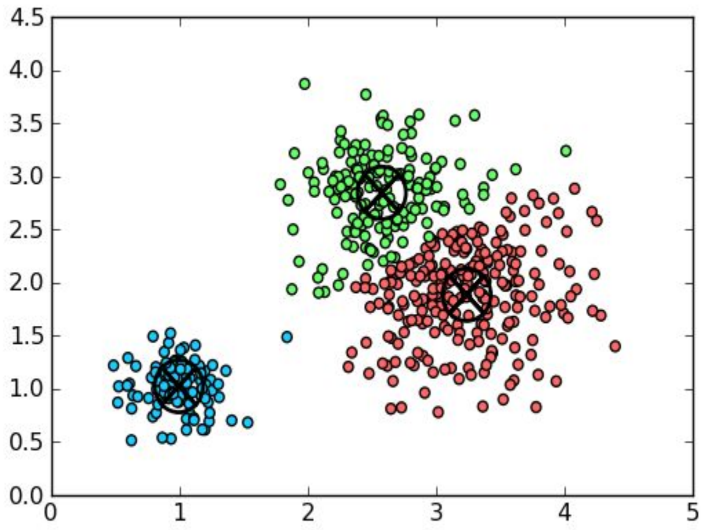
target Output = class label (□ =-1,+=+1)

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

y: <u>Discrete</u> output(s)

UNSUPERVISED LEARNING: Clustering vs. Generative model

Clustering



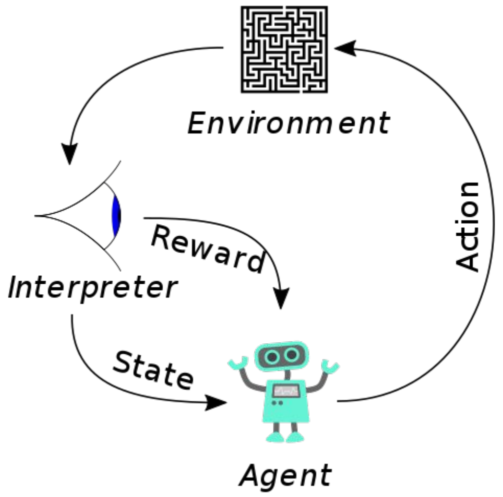
Points = examples
□ partitioning in "groups" (colors) based on similarity

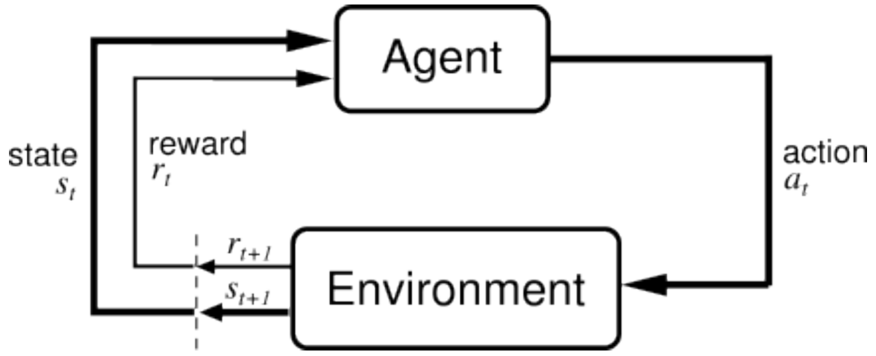
Generative model

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 $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \gamma \in [0,1[$

<u>Typical use of RL: learn a BEHAVIOR</u>

Outline

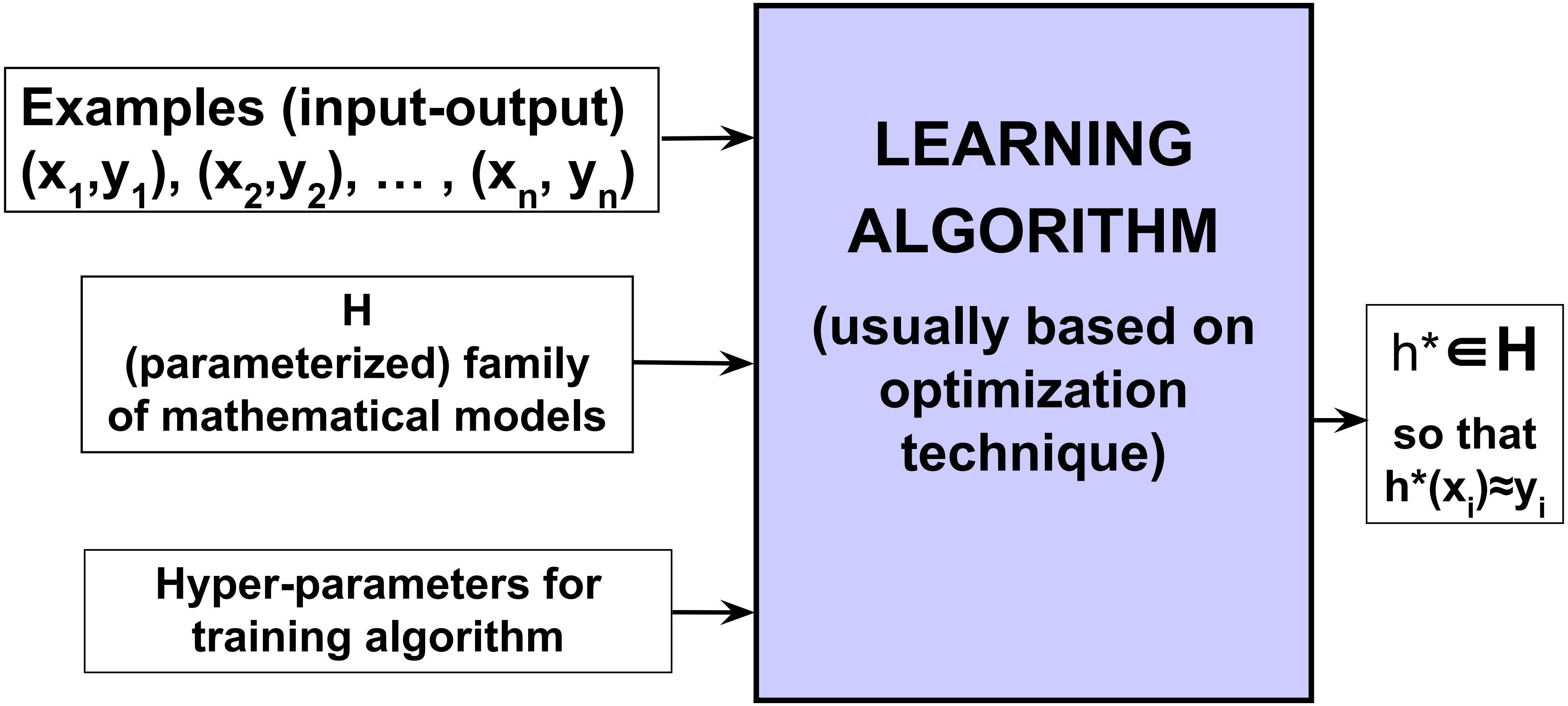
- Intro: What is Statistical Machine-Learning?
- Typology of Machine-Learning
- **General formalism for SUPERVISED Learning**
- Evaluating learnt models:
  metrics for CLASSIFICATION
- Generalization *vs.* overfitting

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,…)
- …

MINES PARIS | PSL

Supervised learning


Examples (input-output)
$(x_1,y_1), (x_2,y_2), \dots , (x_n, y_n)$
H
(parameterized) family of mathematical models
Hyper-parameters for training algorithm
LEARNING ALGORITHM
(usually based on optimization technique)
$h^* \in H$
so that $h^*(x_i) \approx y_i$

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

$K = \Sigma_i \text{ loss}(h(x_i),y_i)$ *[+ 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*"

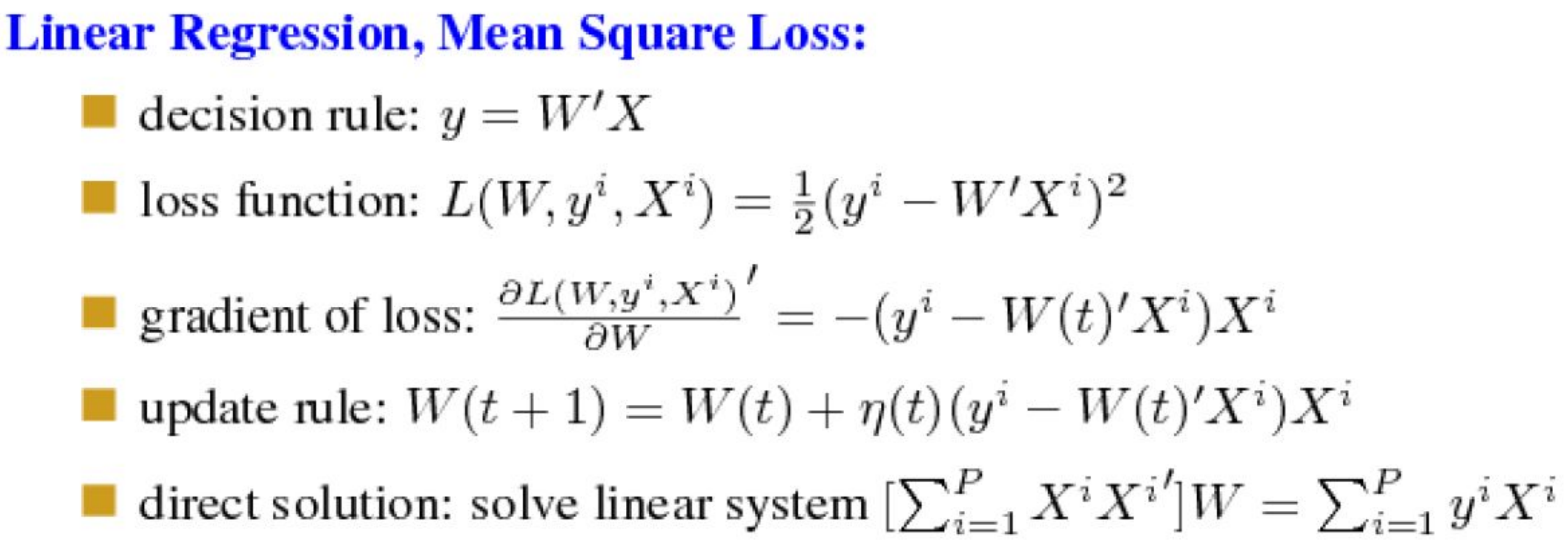
- **The cost function is generally the average over all training examples of a "*loss function*"**

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

(+ 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**

Linear Multivariate Regression



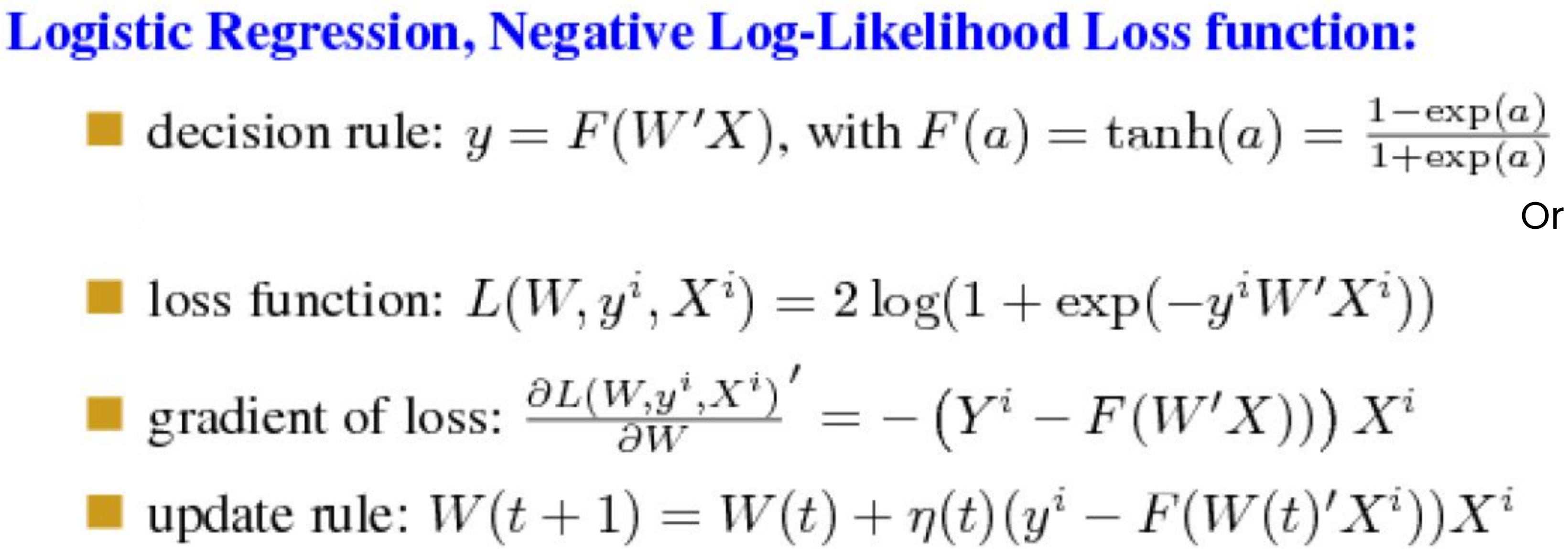
Linear Regression, Mean Square Loss:

- 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:

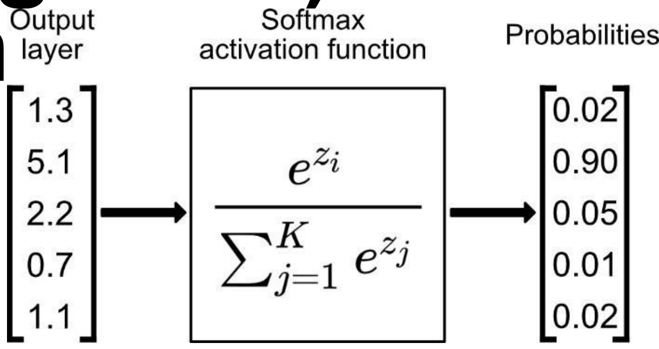
- decision rule: $y = F(W'X)$, with $F(a) = \tanh(a) = \frac{1-\exp(a)}{1+\exp(a)}$

Or Sigmoid

- 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: $W(t+1) = W(t) + \eta(t)(y^i - F(W(t)'X^i))X^i$

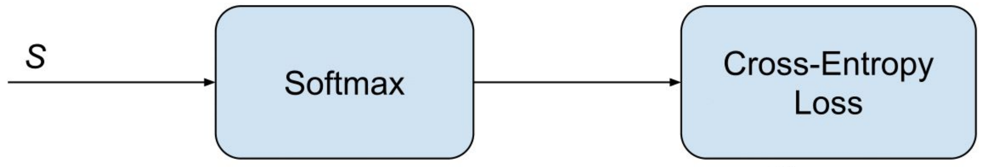
[From slide by Y. LeCun: Machine Learning and Pattern Recognition*]*

Cross Entropy Loss (Log Loss) for Classification

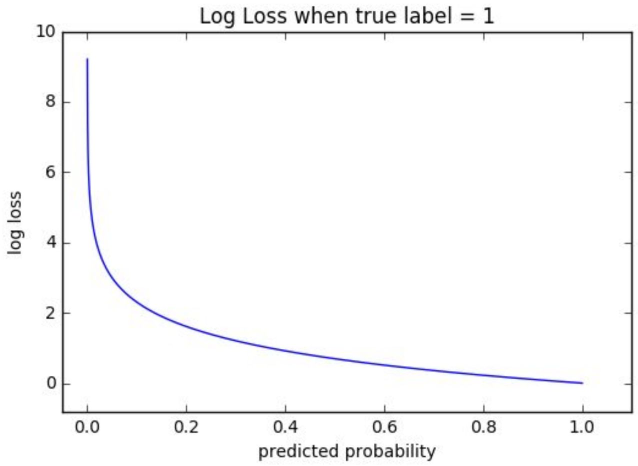


$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- **If** true label 1: Prediction lower than 0.5 is **confident and wrong** answer -> penalize heavily

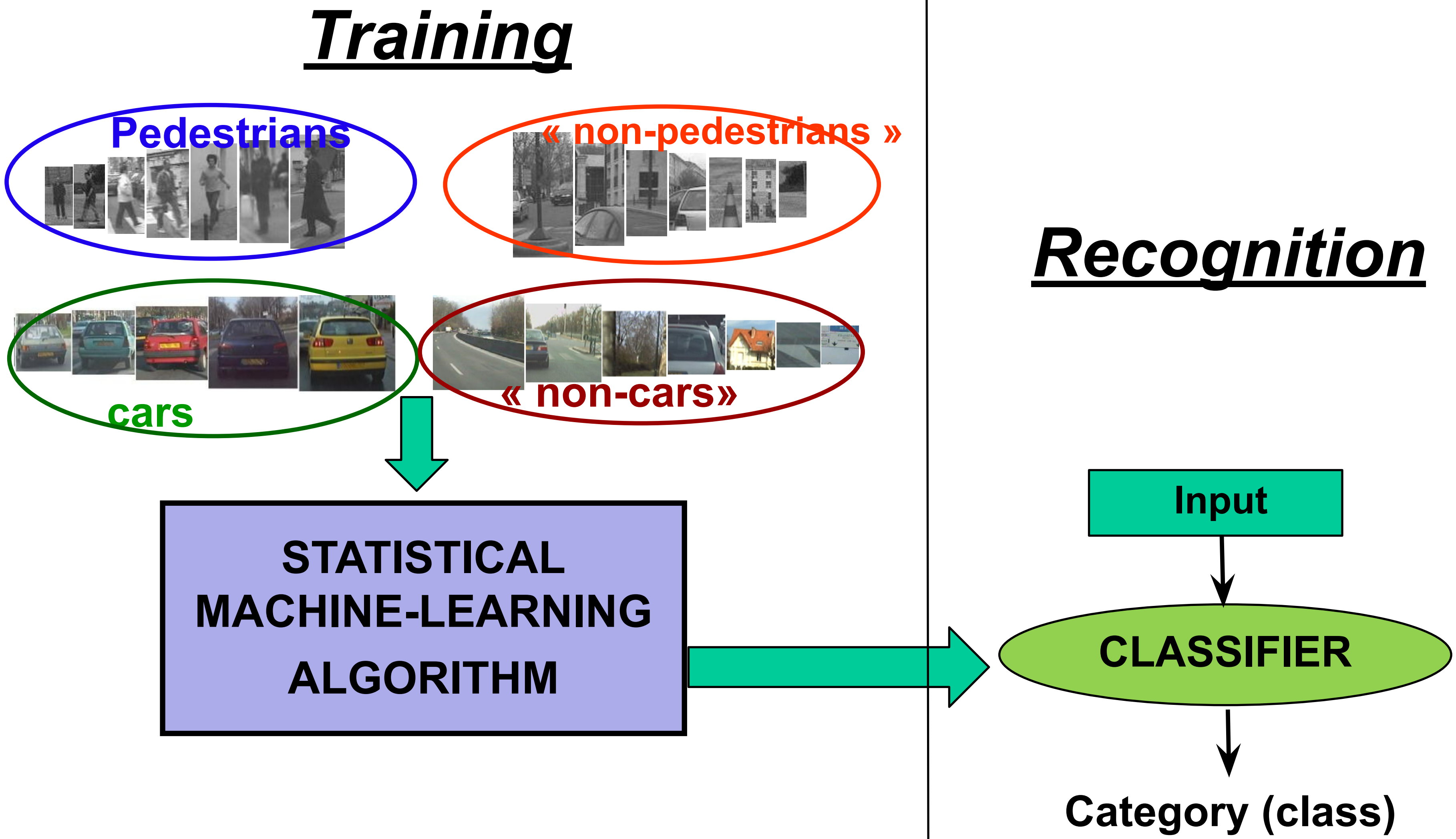


$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$



Compare: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

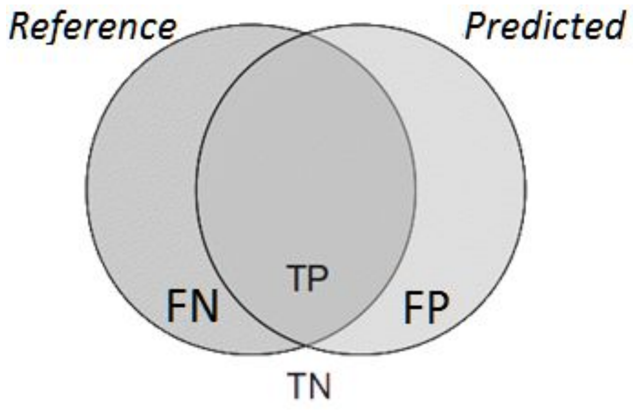
Usual two distinct phases of supervised Machine-Learning



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

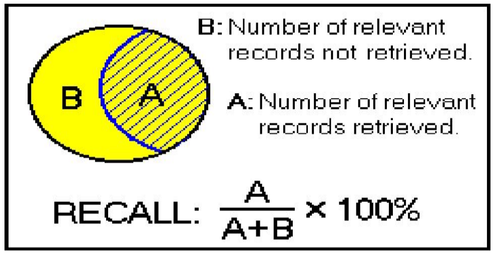


	predicted as positive	predicted as negative
positive	TP	FN
negative	FP	TN

Error rate =

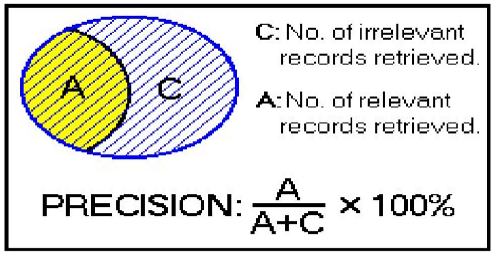
`(FP+FN)/(TP+TN+FP+FN)`

BUT: <u>False Negatives ("missed")</u> ≠ <u>False Positives</u>!


B: Number of relevant records not retrieved.
A: Number of relevant records retrieved.
B
A

$$\text{RECALL: } \frac{A}{A+B} \times 100\%$$

<u>Recall:</u> percentage of relevant examples successfully predicted/retrieved


C: No. of irrelevant records retrieved.
A: No. of relevant records retrieved.
A
C

$$\text{PRECISION: } \frac{A}{A+C} \times 100\%$$

<u>Precision:</u> 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]

$$= \frac{\text{\# of }\underline{\text{correct}}\text{ predictions}}{\text{Total \# of examples}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

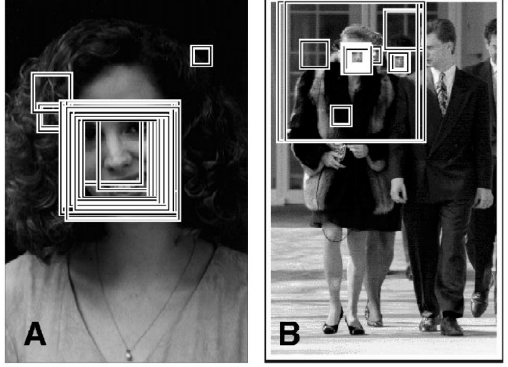
***Recall* (sensitivity)**
True Positive rate

$$= \frac{\text{\# of }\underline{\text{correct }\textit{positive}\text{ }}\text{predictions}}{\text{\# of }\underline{\textit{real}}\text{ positives}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

***Precision* (specificity)**

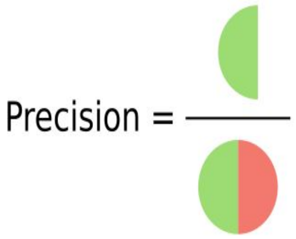
$$= \frac{\text{\# of }\underline{\text{correct }\textit{positive}}\text{ predictions}}{\text{\# of positive }\underline{\textit{predictions}}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

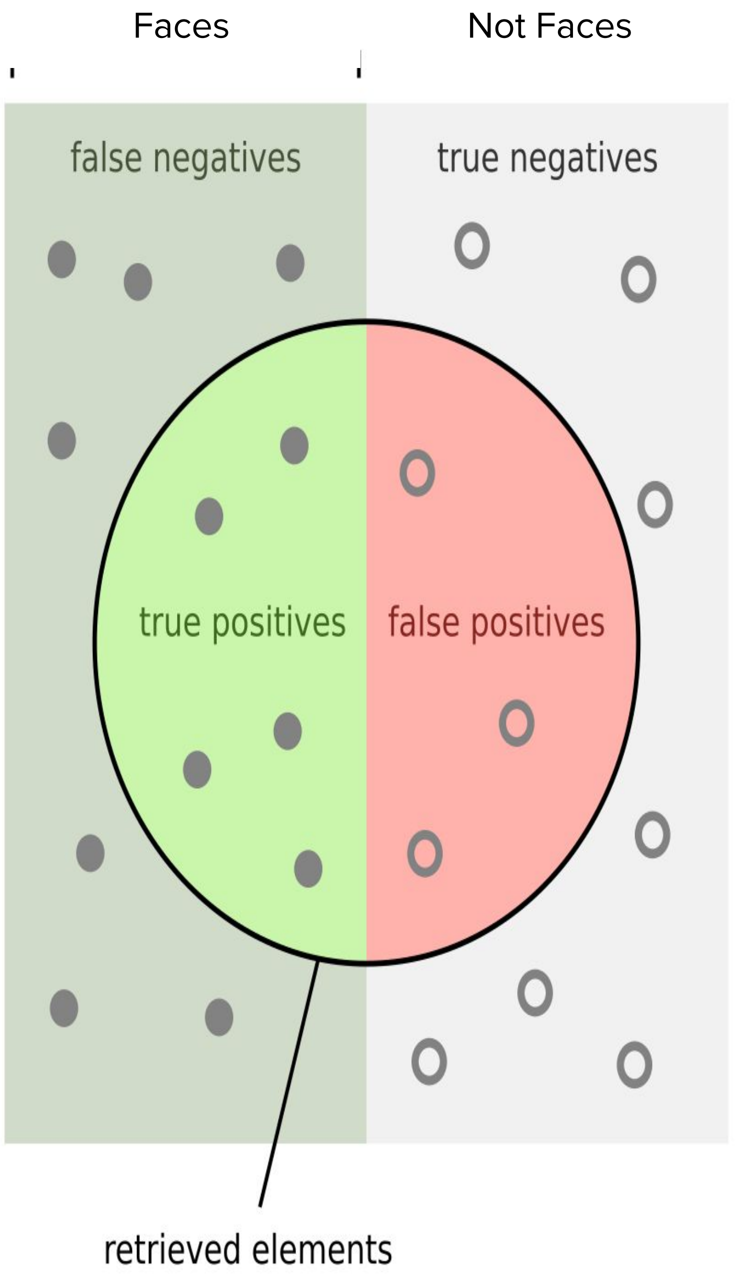
Precision



$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

How many retrieved boxes show faces?





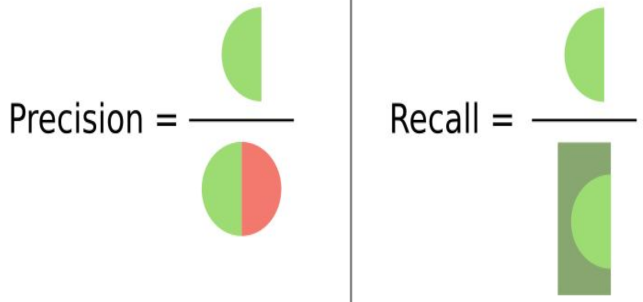
Precision, Recall

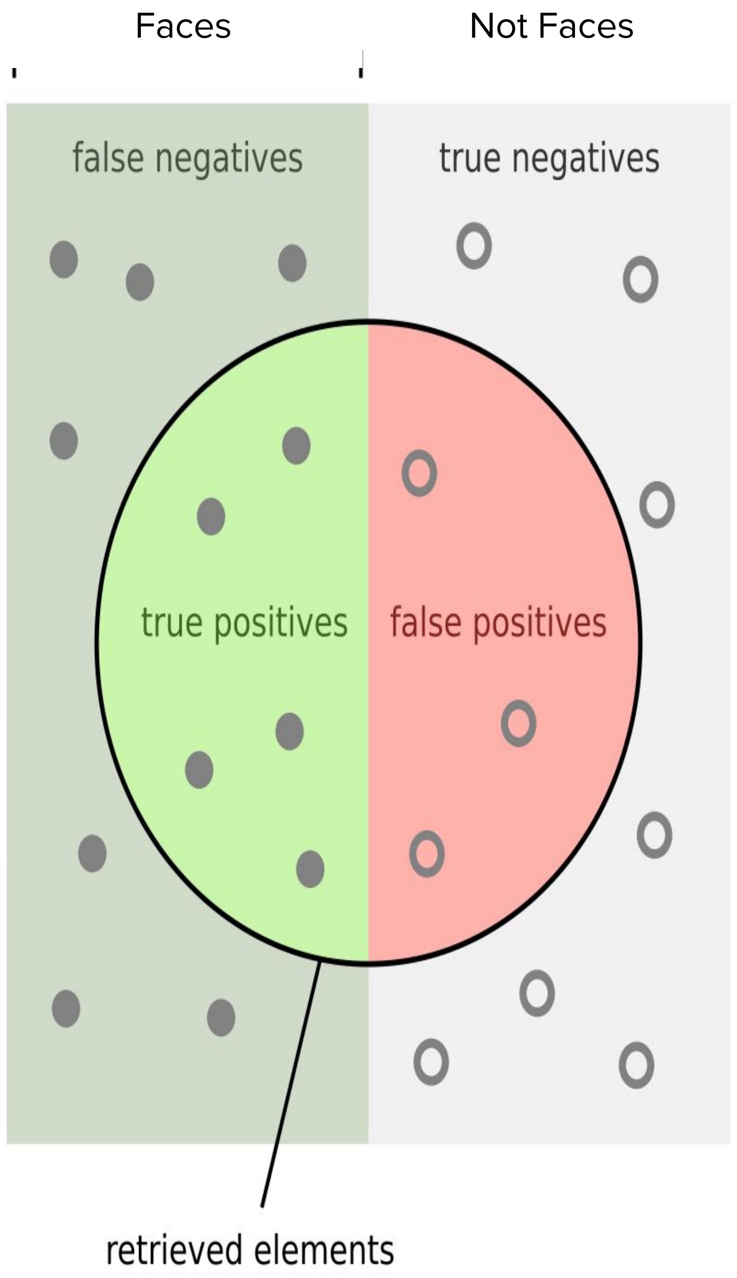
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

How many retrieved boxes show faces?

How many faces are retrieved?





F1 - Score

- Single metric of ‘harmonic mean’
- Mostly used to compare classification results
- Precision and Recall usually inversely related

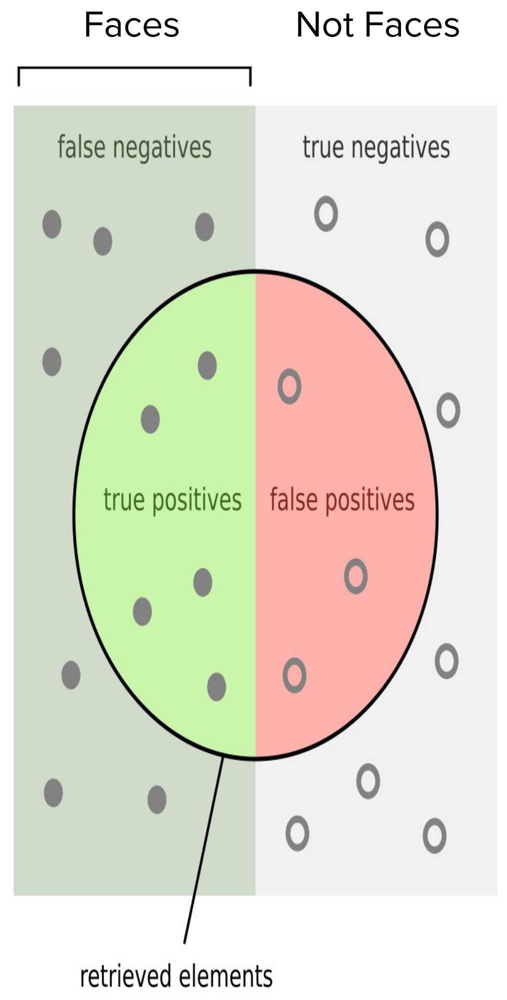
$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Case: High Precision, Low Recall

All boxes contain faces but did not find a lot ->
Classifier too careful, underestimating count of faces

F1 Score -> Numerator goes down due to low recall

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$



How many retrieved boxes show faces?

How many faces are retrieved?

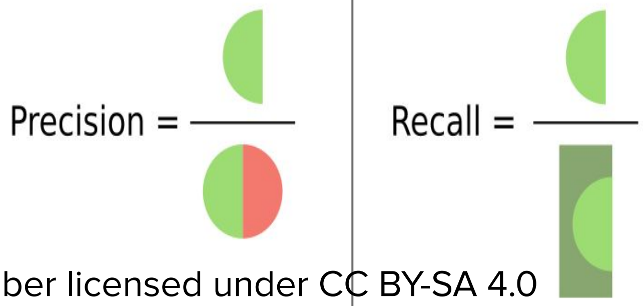


Image by Walber licensed under CC BY-SA 4.0

Other Case: Low Precision, High Recall

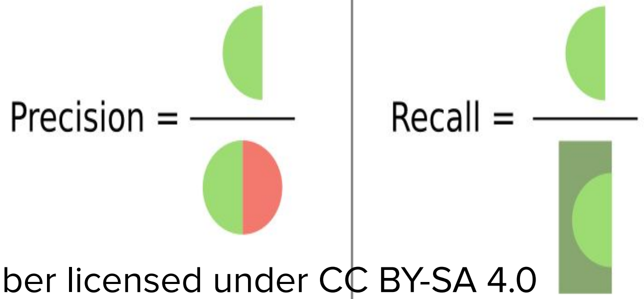
High Recall: No false negatives -> All faces were found
Low Precision: Also, many false positives

Classifier sees faces everywhere
F1 Score -> Numerator goes down due to low precision

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

How many retrieved boxes show faces?

How many faces are retrieved?



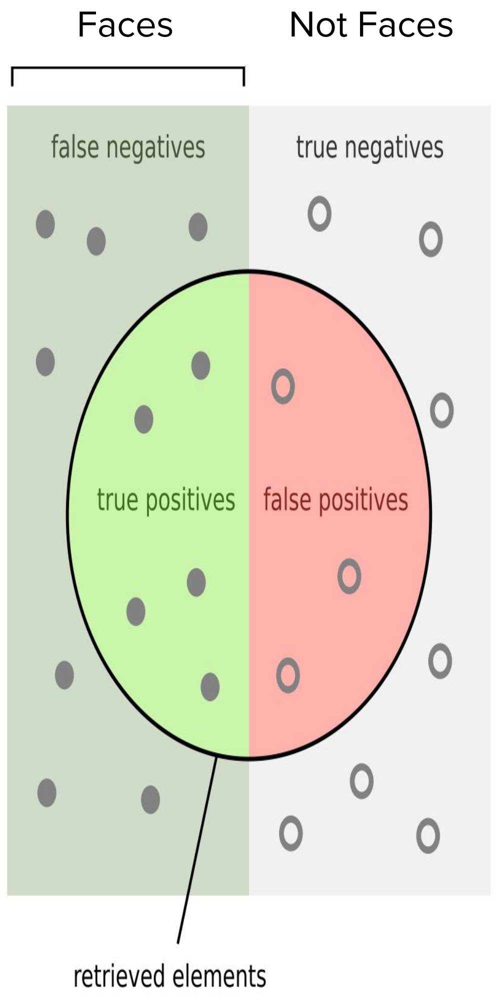


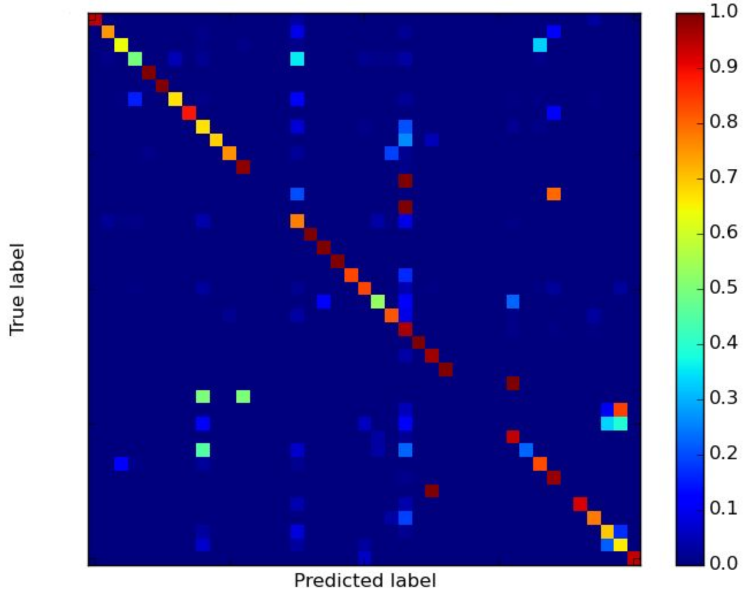
Image by Walber licensed under CC BY-SA 4.0

Classification performance metrics

- <u>Accuracy</u> = proportion of correct
- <u>Recall (sensitivity)</u> ≈ proportion of ”not missed”
  ≈ *”completeness”* level *[exhaustivité]*
- <u>Precision (specificity)</u> ≈ *reliability* of predicted labels
- <u>Confusion matrix</u>: 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%

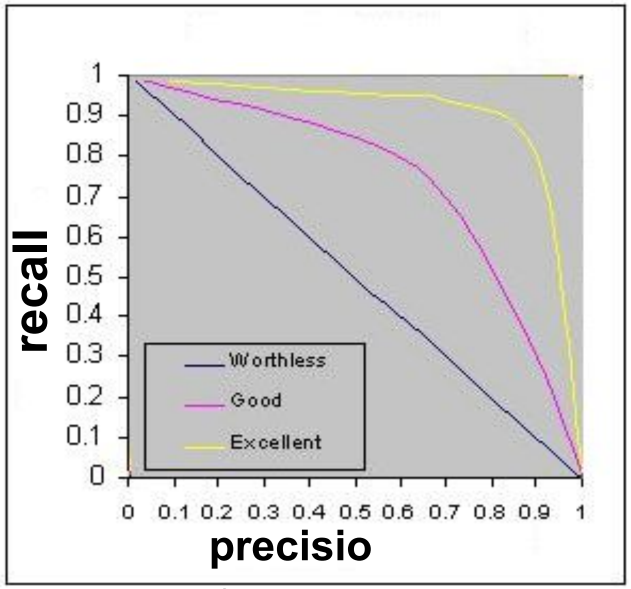


Precision-recall trade-off and curve

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* »

  often $= \|h(x)-y\|^2$

- **What optimum for h?**

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

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

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

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

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

$$E(h^*_{H,n}) - E(h^*) = \underbrace{[E(h^*_{H,n}) - E(h^*_H)]}_{\textit{ESTIMATION error}} + \underbrace{[E(h^*_H) - E(h^*)]}_{\textit{MODEL error}}$$

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 <u>*+ GENERALIZE*</u>"

Given a <u>FINITE</u> 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 $y_i = f(x_i) + \text{noise}$

⇒ goal = to <u>minimize the GENERALIZATION error</u>

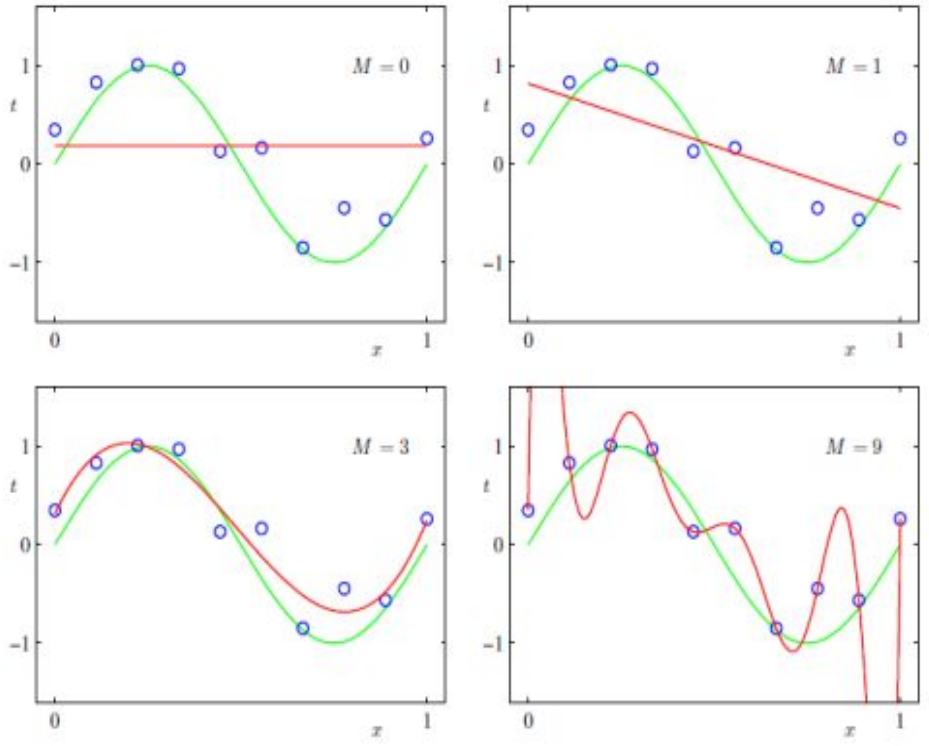
$$E_{gen} = \int \| h(x) - f(x) \|^2 \, p(x)\, dx$$

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

About over-fitting

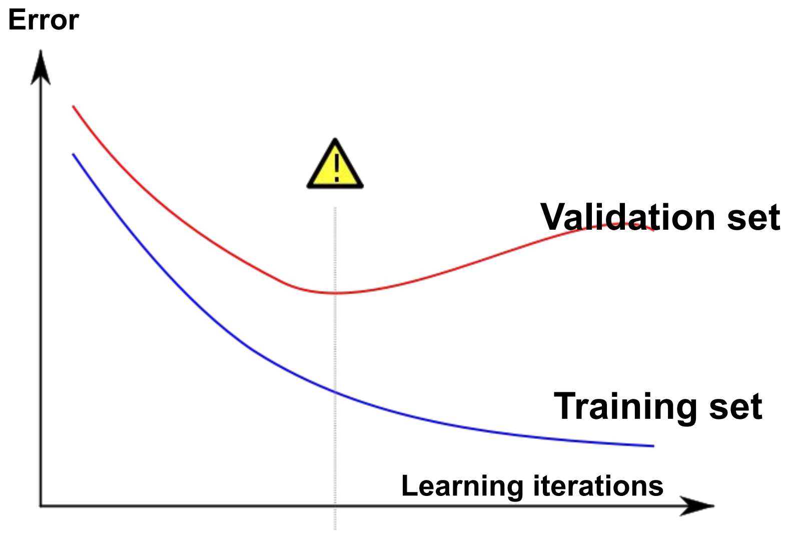
The generalization error cannot be directly measured, only *empirical error* on examples can be estimated:

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



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

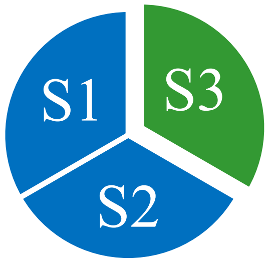
Detection of over-fitting
for an iterative algorithm



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

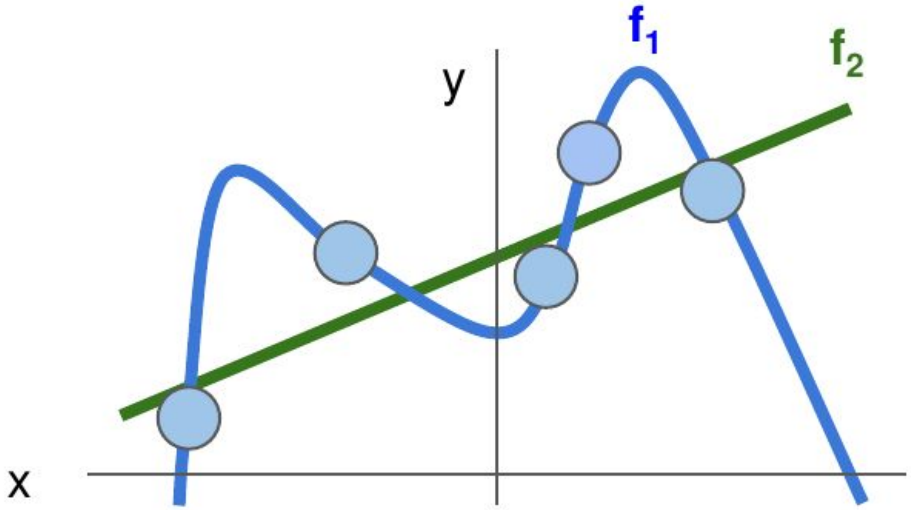
Regularization

Want to find:

- "Simplest" h
- Often, add term to loss
- Consider:
  - Loss L
  - Weights W
  - Function f

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{}$$

Data loss: Model predictions should match training data

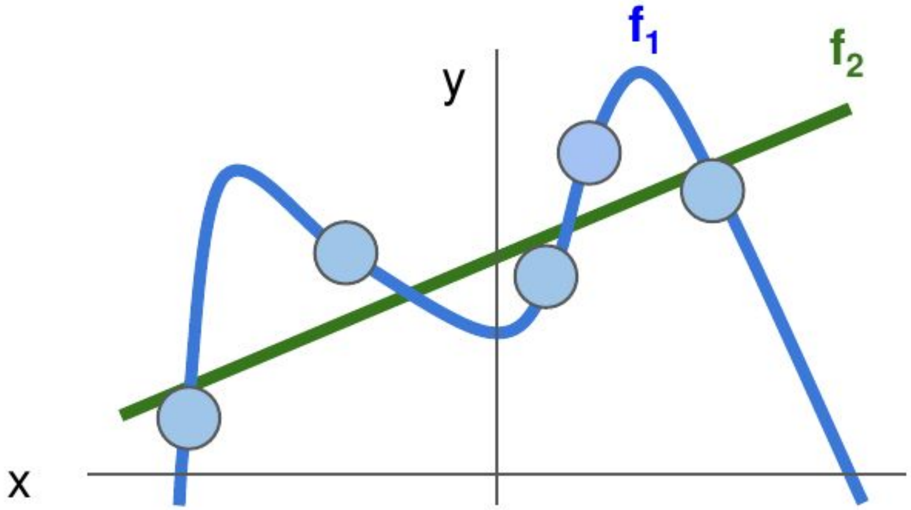


From: Lecture 3: Regularization and Optimization, Fei-Fei Li, Ehsan Adeli, Zane Durante

Regularization

Want to find:

- "Simplest" h
- Often, add term to loss
- Consider:
  - Loss L
  - Weights W
  - Function f
  - Regularization R



$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{\textbf{Data loss}: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{\textbf{Regularization}: Prevent the model from doing \textit{too} well on training data}}$$

Regularization

Want to find:

- "Simplest" h
- Often, add term to loss
- Consider:
  - Loss L
  - Weights W
  - Function f
  - Regularization R

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Prevent the model from doing too well on training data}}$$

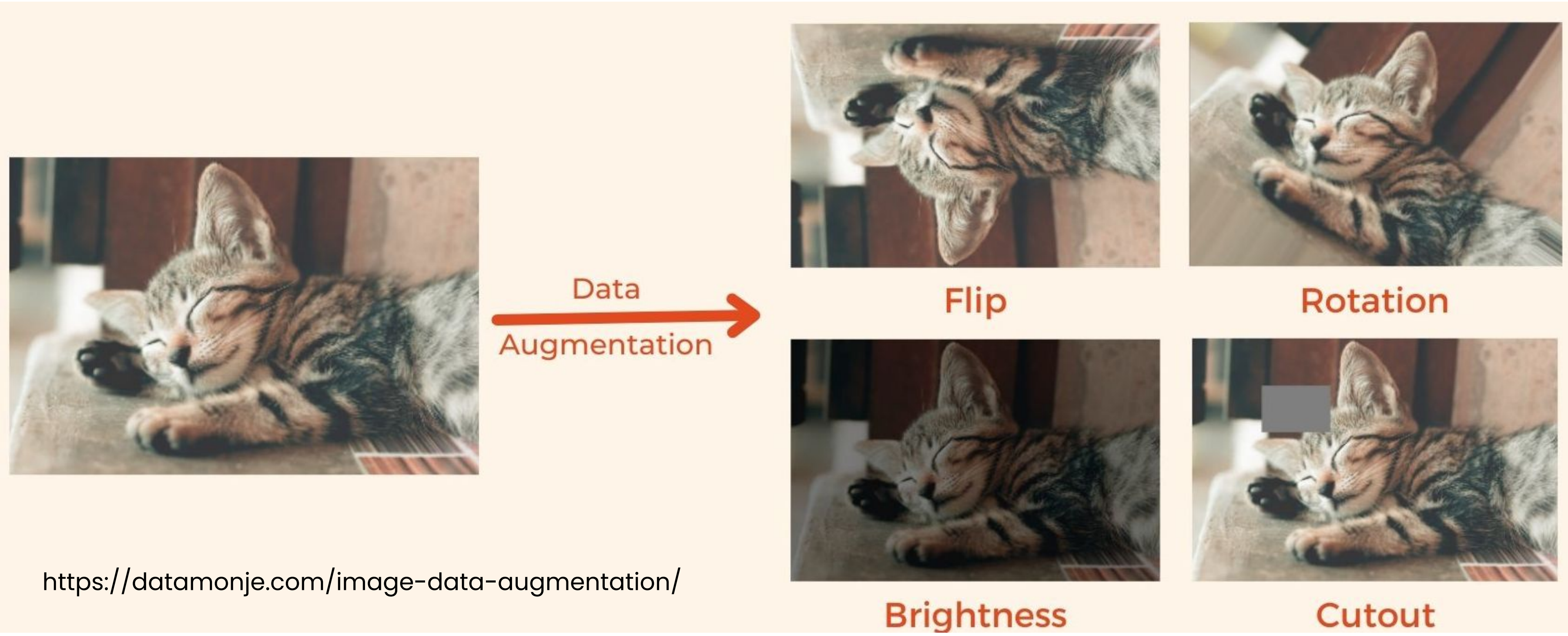
Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Data augmentation (for classification)

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

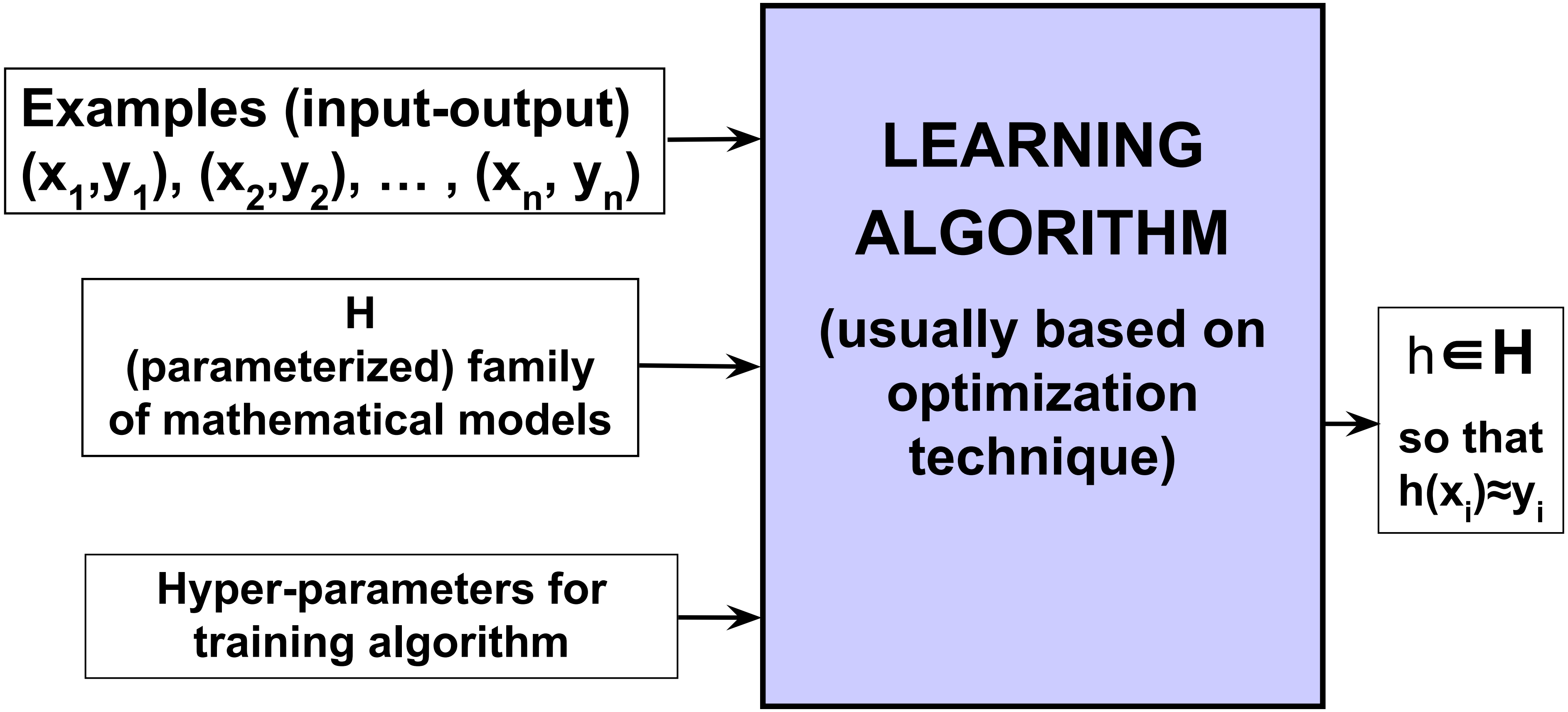
- for each labelled example in training set, generate several slightly *distorted* variants which shall have the same label



https://datamonje.com/image-data-augmentation/

Synthesis on various algorithms for SUPERVISED Machine-Learning

Supervised learning


Examples (input-output)
$(x_1,y_1), (x_2,y_2), \dots , (x_n, y_n)$
H
(parameterized) family of mathematical models
Hyper-parameters for training algorithm
LEARNING ALGORITHM
(usually based on optimization technique)
$h \in H$
so that $h(x_i) \approx y_i$

Typology of *classification* methods

- 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

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

MINES PARIS | PSL

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-lear ning)	-			
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.