MACHINE LEARNING

Liviu Ciortuz
Department of CS, University of Iaşi, România

What is Machine Learning?

- **ML studies algorithms that $\underbrace{\text{improve with}}_{\text{learn from}}$ experience.**

  Tom Mitchell's **Definition of the [*general*] *learning problem*:**
  "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 on tasks in T, as measured by P, improves with experience E."

- Examples of [specific] learning problems (see next slide)
- [Liviu Ciortuz:] ML is data-driven programming
- [Liviu Ciortuz:] ML gathers a number of well-defined sub-domains/disciplines, each one of them aiming to solve in its own way the above-formulated [*general*] *learning problem*.

What is Machine Learning good for?

- natural language (text & speech) processing
- genetic sequence analysis
- robotics
- customer (financial risc) evaluation
- terrorist threat detection
- compiler optimisation
- semantic web
- computer security
- software engineering
- computer vision (image processing)
- etc.

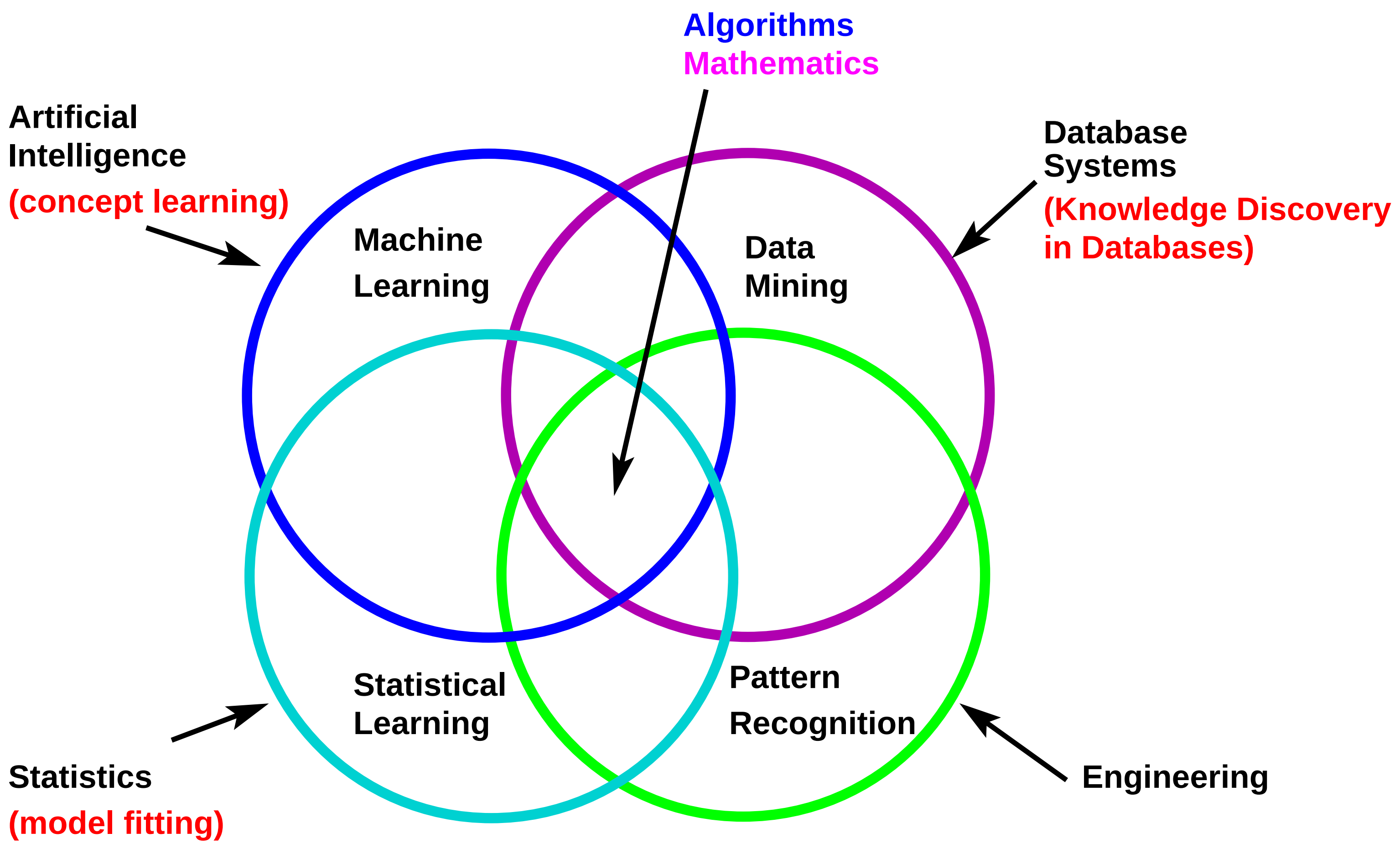
Related courses at FII

- Artificial Intelligence
- Genetic Algorithms
- Artificial Neural Networks
- Probabilistic programming

- Special Chapters of Machine Learning
- Special Chapters of Artificial Intelligence
- Special Chapters of Artificial Neural Networks
- Data Mining
- Nature-inspired computing methods
- Big Data Analytics
- Image Processing
- Computer Vision

◦ Bioinformatics

A multi-domain view



The Machine Learning Undergraduate Course: Plan

0. Introduction to Machine Learning (T. Mitchell, ch. 1)

---

1. Probabilities Revision (Ch. Manning & H. Schütze, ch. 2)

---

2. Decision Trees (T. Mitchell, ch. 3)
3. Bayesian Learning (T. Mitchell, ch. 6)
   [and the relationship with Logistic Regression]
4. Instance-based Learning (T. Mitchell, ch. 8)
5. Clustering Algorithms (Ch. Manning & H. Schütze, ch. 14)

The Machine Learning Master Course:

Tentative Plan

1. Decision Trees: Boosting
2. Support Vector Machines (N. Cristianini & J. Shawe-Taylor, 2000)
3. Computational Learning Theory (T. Mitchell, ch. 7)

Probabilities Revision (Ch. Manning & H. Schütze, ch. 2)

4. Gaussian Bayesian Learning
5. The EM algorithmic schemata (T. Mitchell, ch. 6.12)
6. Hidden Markov Models (Ch. Manning & H. Schütze, ch. 9)

Bibliography

0. “Exerciţii de învăţare automată”
L. Ciortuz, A. Munteanu E. Bădărău.
Iaşi, Romania, 2023
www.info.uaic.ro/∼ciortuz/ML.ex-book/editia-2023f/ex-book.20sept2023.pdf

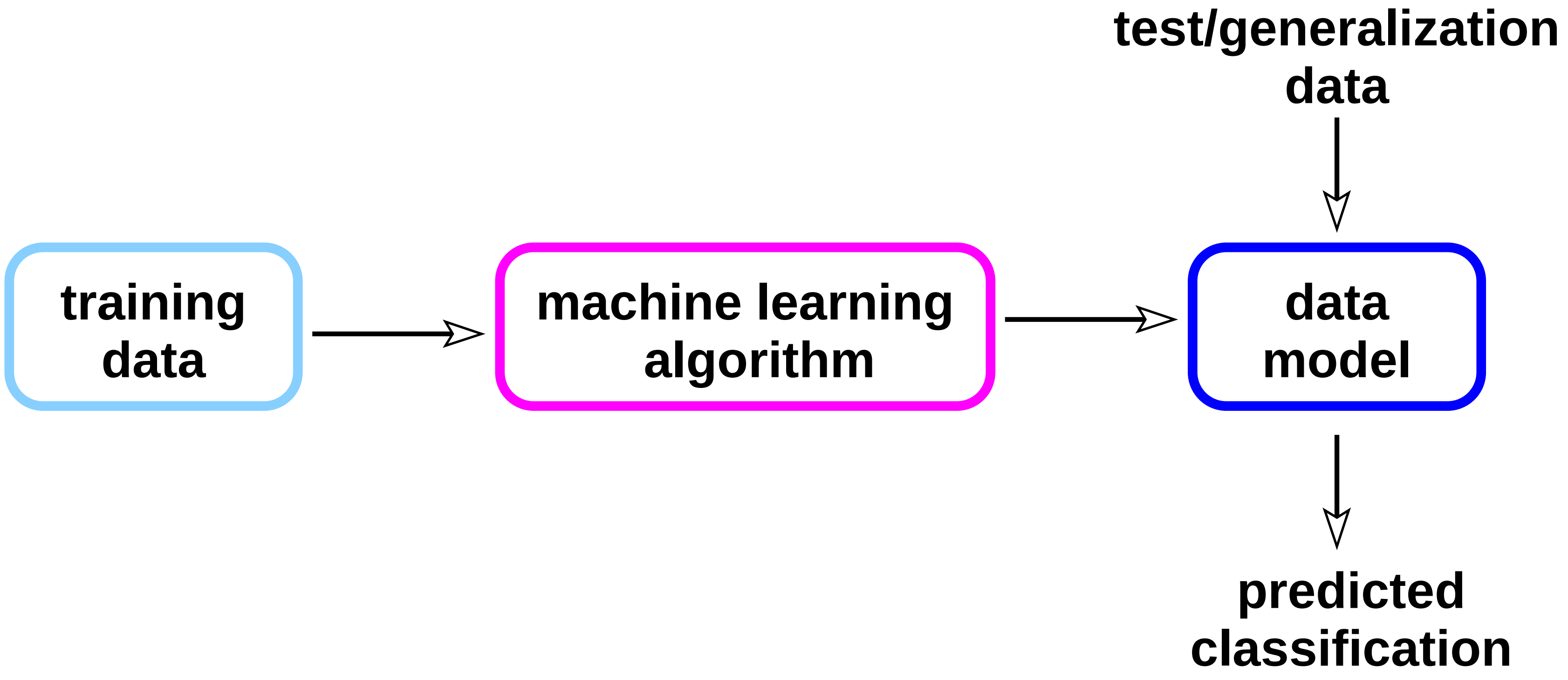
---

1. “Machine Learning”
Tom Mitchell. McGraw-Hill, 1997
2. “Machine Learning Foundations”
Teaho Jo. Springer, 2021
3. “Deep Machine Learning Foundations”
Teaho Jo. Springer, 2023
4. “Foundations of Statistical Natural Language Processing”
Christopher Manning, Hinrich Schütze. MIT Press, 2002

---

5. “Support Vector Machines and other kernel-based learning methods”
Nello Cristianini, John Shawe-Taylor. Cambridge University Press, 2000.

A general schema for machine learning methods



“We are drawning in information but starved for knowledge.”

John Naisbitt, “Megatrends” book, 1982

Basic ML Terminology

1. **instance x, instance set X**
   concept $c \subseteq X$, or $c : X \rightarrow \{0,1\}$
   example (labeled instance): $\langle x, c(x) \rangle$; positive examples, neg. examples
2. **hypotheses $h : X \rightarrow \{0,1\}$**
   hypotheses representation language
   hypotheses set H
   hypotheses consistent with the concept c: $h(x) = c(x), \forall$ example $\langle x, c(x) \rangle$
   version space
3. **learning = train + test**
   supervised learning (classification), unsupervised learning (clustering)
4. ***error***$_h = \mid \{x \in X, h(x) \neq c(x)\} \mid$
   training error, test error
   accuracy, precision, recall
5. **validation set, development set**
   n-fold cross-validation, leave-one-out cross-validation
   overfitting

The Inductive Learning Assumption

Any hypothesis found to conveniently approximate the target function over a sufficiently large set of training examples

will also conveniently approximate the target function over other unobserved examples.

Inductive Bias

Consider

- a concept learning algorithm L
- the instances X, and the target concept c
- the training examples $D_c = \{\langle x, c(x)\rangle\}$.
- Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on data D_c.

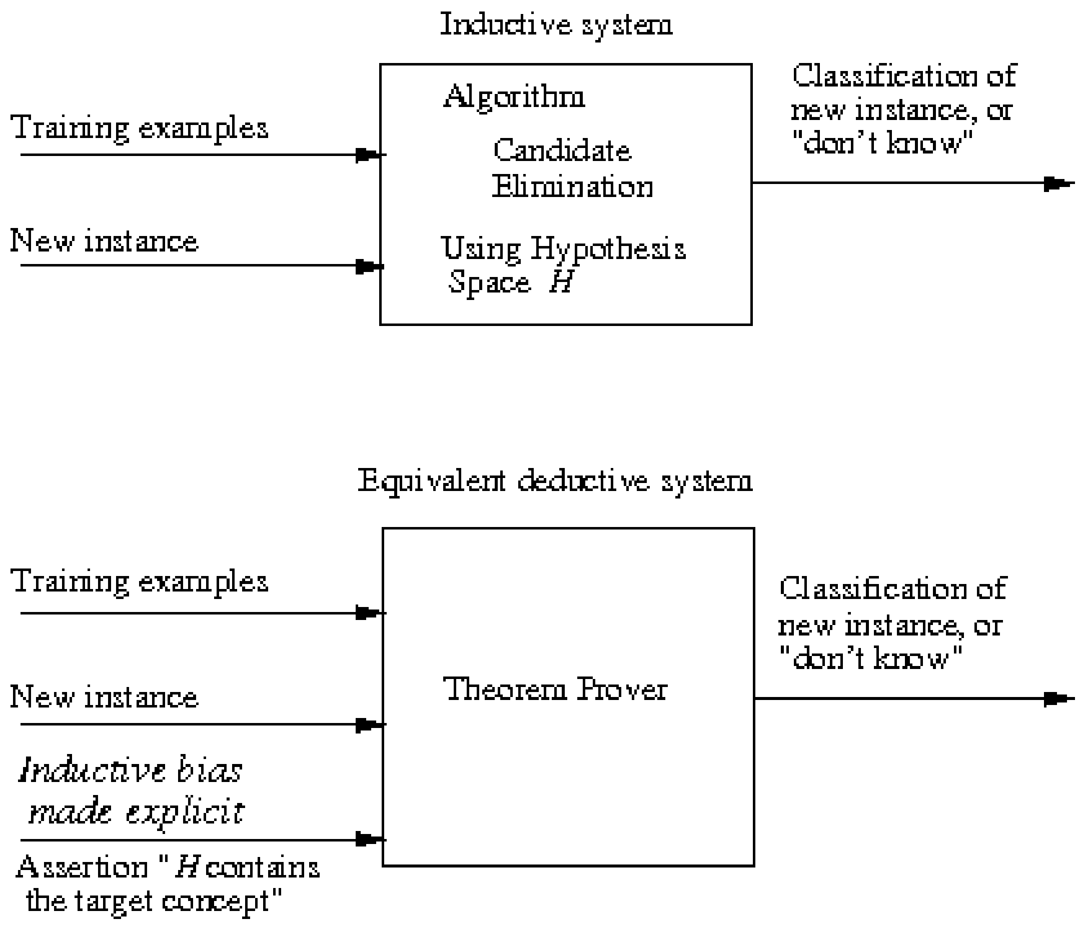
Definition:

The inductive bias of L is any minimal set of assertions B such that

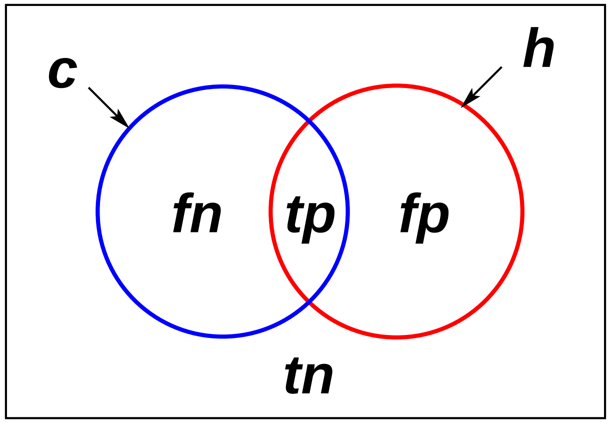
$$(\forall x_i \in X)[(B \vee D_c \vee x_i) \vdash L(x_i, D_c)]$$

for any target concept c and corresponding training examples D_c.
($A \vdash B$ means A logically entails B)

Inductive systems can be modelled by equivalent deductive systems


Inductive system
Training examples
New instance
Algorithm
Candidate Elimination
Using Hypothesis Space *H*
Classification of new instance, or "don't know"
Equivalent deductive system
Training examples
New instance
Inductive bias made explicit
Assertion "*H* contains the target concept"
Theorem Prover
Classification of new instance, or "don't know"

Evaluation measures in Machine Learning



tp	–	true positives
fp	–	false positives
tn	–	true negatives
fn	–	false negatives

accuracy: $Acc = \frac{tp + tn}{tp + tn + fp + fn}$

precision: $P = \frac{tp}{tp + fp}$

recall (or: sensitivity): $R = \frac{tp}{tp + fn}$

F-measure: $F = \frac{2\ P \times R}{P+R}$

specificity: $Sp = \frac{tn}{tn + fp}$

follout: $= \frac{fp}{tn + fp}$

Mathew's Correlation Coefficient:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)\times(tn + fn)\times(tp + fn)\times(tn + fp)}}$$

Lazy learning vs. eager learning algorithms

Eager: generalize before seeing query

- ◦ ID3, Backpropagation, Naive Bayes, Radial basis function networks, . . .
- Must create global approximation

Lazy: wait for query before generalizing

- ◦ k-Nearest Neighbor, Locally weighted regression, Case based reasoning
- Can create many local approximations

Does it matter?

If they use the same hypothesis space H, lazy learners can represent more complex functions.

E.g., a lazy Backpropagation algorithm can learn a NN which is different for each query point, compared to the eager version of Backpropagation.

Basic Machine Learning Algorithms

ID3 algorithm: a simplified version

Ross Quinlan, 1979, 1986

```
START
    create the root $node$;
    assign all examples to the root node;
Main loop:
  1. $A \leftarrow$ the “best” decision attribute for the next $node$;
  2. for each value of $A$, create a new descendant of $node$;
  3. sort training examples to leaf nodes;
  4. if training examples are perfectly classified, then
     STOP;
     else iterate over the new leaf nodes
```

AdaBoost algorithm [Yoav Freund, Robert Schapire, 1996, 1997, 1999]

Consider m training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$. Suppose we have a *weak learning algorithm* A which produces a hypothesis $h : \mathcal{X} \to \{-1, +1\}$ given any distribution D of examples.

- **Begin with a uniform distribution $D_1(i) = \frac{1}{m}$, $i = 1, \ldots, m$.**
- **At each iteration $t = 1, \ldots, T$,**
  - **run the weak learning algo A on the distribution D_t and produce the hypothsis h_t;**

  ***Note* (1): Since A is a weak learning algorithm, the produced hypothesis h_t at round t is only slightly better than random guessing, say, by a margin γ_t:**

  $$\varepsilon_t = err_{D_t}(h_t) = \Pr_{x \sim D_t}[y \neq h_t(x)] = \frac{1}{2} - \gamma_t.$$

  ***Note* (2): If at a certain iteration $t \leq T$ the weak classifier A cannot produce a hypothesis better than random guessing (i.e., $\gamma_t = 0$) or it produces a hypothesis for which $\varepsilon_t = 0$, then the AdaBoost algorithm should be stopped.**

  - **update the distribution**

  $$D_{t+1}(i) = \frac{1}{Z_t} \cdot D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)} \text{ for } i = 1, \ldots, m, \qquad (1)$$

  where $\alpha_t \overset{not.}{=} \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$, and Z_t is the normalizer.
- **In the end, deliver $H_T = sign\left(\sum_{t=1}^{T} \alpha_t h_t\right)$ as the learned hypothesis, which will act as a *weighted majority vote*.**

AdaBoost as an instance of a more general *stepwise algorithm*

Input: S, T, $\mathcal{H}$, ϕ, **where**

$S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ **is the training dataset, with** $y_i \in \{-1, +1\}$
T **is the number of iterations to be executed,**
$\mathcal{H}$ **is a set of "hypotheses",**
$\phi(y, y')$ **is a "loss" / "cost" / "risk" function;**

Procedure:

```
Initialize the classifier by taking $f_0(x) = 0$ (the constant function $0$),
        and $D_1(i) = 1/m$ for $i = 1, \ldots, m$
for $t = 1$ to $T$ do:
    1. Compute
        $(h_t, \alpha_t) = \arg\min_{\alpha \in \mathbb{R}, h \in \mathcal{H}} \sum_{i=1}^{m} \phi(y_i, f_{t-1}(x_i) + \alpha h(x_i))$
    2. Update the classifier
        $f_t(x) = f_{t-1}(x) + \alpha_t h_t(x)$
        compyte $D_{t+1}$
end for
return the classifier *sign* $(f_T(x))$
```

***Note*: At each step, the algorithm greedily adds a hypothesis $h \in \mathcal{H}$ to the current *combined hypothesis* to minimize the ϕ-loss.**

A Generalized AdaBoost Algorithm

[MIT, 2003 fall, Tommi Jaakkola, HW4, pr. 2.1-3]

Initialization: $\tilde{W}_i^{(1)} = 1/m$ **and** $f_0(x_i) = 0$ **for** $i = 1, \ldots, m$.

Loop: **for** $t = 1$ **to** T **do:**

***Step 1*:** **Find a classifier** $h(x; \hat{\theta}_t)$ **performing better than chance wrt the weighted training error:**

$$\varepsilon_t \overset{not.}{=} \sum_{i \,:\, y_i \neq h(x_i; \hat{\theta}_t)} \tilde{W}_i^{(t)} y_i h(x_i; \theta) = \frac{1}{2}\left(1 - \sum_{i=1}^m \tilde{W}_i^{(t)} y_i h(x_i; \hat{\theta}_t)\right).$$

Note: **Minimizing** ε_t **is equivalent to finding** $\hat{\theta}_t$ **that minimizes** $\frac{\partial}{\partial \alpha} J_t(\alpha, \theta_t)_{|\alpha=0}$ **wrt** θ_t**, where**

$$J_t(\alpha, \theta_t) = \frac{1}{m} \sum_{i=1}^m \text{Loss}(y_i\, f_{t-1}(x_i) + y_i\, \alpha\, h(x_i; \theta_t)).$$

***Step 2*:** **Set the votes** α_t **for the new component by minimizing the overall empirical loss:**

$$\alpha_t \quad = \quad \arg\min_{\alpha \geq 0} J_t(\alpha, \hat{\theta}_t).$$

***Step 3*:** **Recompute the normalized weights for the next iteration according to**

$$\tilde{W}_i^{(t+1)} = -c_t \cdot dL(\underbrace{y_i\, f_{t-1}(x_i) + y_i\, \alpha_t\, h(x_i; \hat{\theta}_t)}_{y_i\, f_t(x_i)}) \text{ for } i = 1, \ldots, m,$$

where c_t **is chosen so that** $\sum_{i=1}^m \tilde{W}_i^{(t+1)} = 1$.

Output: f_T

The Naive Bayes Classifier

- **Assume that the attributes $< a_1, \ldots, a_n >$ that describe instances are conditionally independent w.r.t. to the given classification:**

$$P(a_1, a_2 \ldots a_n | v_j) = \prod_i P(a_i | v_j)$$

- **Training procedure:**

```
NAIVE_BAYES_LEARN(examples)
    for each value $v_j$ of the output attribute
        $\hat{P}(v_j) \leftarrow$ estimate $P(v_j)$
        for each value $a_i$ of each input attribute $a$
            $\hat{P}(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$
```

- **The *decision rule* of the Naive Bayes classifier is:**

$$\begin{aligned} v_{MAP} &= \operatorname*{argmax}_{v_j \in V} P(v_j | a_1, a_2 \ldots a_n) = \operatorname*{argmax}_{v_j \in V} \frac{P(a_1, a_2 \ldots a_n | v_j) P(v_j)}{P(a_1, a_2 \ldots a_n)} \\ &= \operatorname*{argmax}_{v_j \in V} P(a_1, a_2 \ldots a_n | v_j) P(v_j) = \operatorname*{argmax}_{v_j \in V} \prod_i P(a_i | v_j) P(v_j) \overset{not.}{=} v_{NB} \end{aligned}$$

Logistic Regression

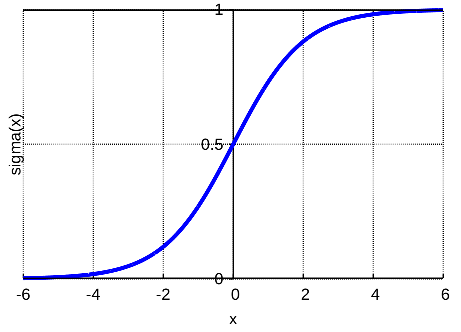
$$P(Y=1|X=x)=\sigma(z) \Leftrightarrow$$
$$P(Y=0|X=x)=1-\sigma(z),$$ **where**

$$\sigma(z) \overset{def.}{=} \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z},$$

$$z \overset{not.}{=} w_0 + \sum_{i=1}^{d} w_i x_i \overset{not.}{=} w \cdot x,$$ **with**

$$w \overset{not.}{=} (w_0, w_1, \ldots, w_d) \in \mathbb{R}^{d+1},$$

assuming $x_0 = 1$.



Given the dataset $\{(x^{(1)}, y^{(1)}),\ (x^{(2)}, y^{(2)}), \ldots,\ (x^{(n)}, y^{(n)})\}$**, where each vector** $x^{(i)}$ **has** d **features / attributes, and** $y^{(i)} \in \{0,1\}$ **for** $i = 1, \ldots, n$**, its complete *log-likelihood* is:**

$$\begin{aligned}
\textit{log-likelihood} &= \ln \prod_{i=1}^{n} P(x^{(i)}, y^{(i)}) = \ln \prod_{i=1}^{n} (P_{Y\,|\,X}(y^{(i)}|x^{(i)})\, P_X(x^{(i)})) \\
&= \ln \left(\left(\prod_{i=1}^{n} P_{Y\,|\,X}(y^{(i)}|x^{(i)}) \right) \cdot \left(\prod_{i=1}^{n} P_X(x^{(i)}) \right) \right) \\
&= \ln \prod_{i=1}^{n} P_{Y\,|\,X}(y^{(i)}|x^{(i)}) + \ln \prod_{i=1}^{n} P_X(x^{(i)}) \overset{not.}{=} \ell(w) + \ell_x .
\end{aligned}$$

Note that ℓ_x **does not depend on the parameter** w**.**

It can be shown that the conditional *log-likelihood* function $\ell(w)$ **can be written as:**

$$\ell(w) = \sum_{i=1}^{n} \left(y^{(i)} \ln \sigma(w \cdot x^{(i)}) + (1-y^{(i)}) \ln(1-\sigma(w \cdot x^{(i)})) \right)$$

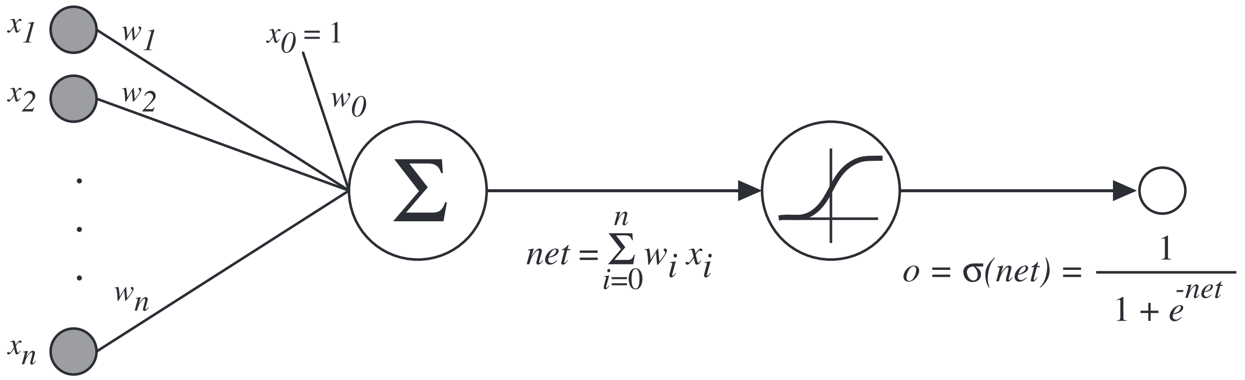
$$w_{LogR} \overset{def.}{=} \underset{w}{\text{argmax}}\, \ell(w) = \arg\min_{w}(-\ell(w)). \qquad (2)$$

Note that $-\ell(w)$ **is a cross-entropy.**

For a more general result than (2)**, see Stanford, 2015 fall, Andrew Ng, HW 3, pr. 5.c.**

See the analogy with the sigmoidal perceptron

CMU, 2011 fall, Eric Xing, HW1, pr. 3.3


x_1
x_2
x_n
w_1
w_2
w_n
$x_0 = 1$
w_0
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1 + e^{-net}}$

The k-Nearest Neighbor Argorithm

Evelyn Fix, Joseph Hodges, 1951; Thomas Cover, Peter Hart, 1967

Training:

Store all training examples.

Classification:

Given a query/test instance x_q,
first locate the k nearest training examples $x_1, \ldots, x_k$,
then estimate $\hat{f}(x_q)$:

- **take a vote among its k nearest neighbors**

$$\hat{f}(x_q) \leftarrow \operatorname*{argmax}_{v \in V} \sum_{i=1}^{k} 1_{\{f(x_i)=v\}}$$

where $1_{\{\,\cdot\,\}}$ is the well-known *indicator function*.

The Bottom-up Hierarchical Clustering Algorithm

```
Given: a set $X = \{x_1, \ldots, x_n\}$ of objects
       a function sim: $\mathcal{P}(X) \times \mathcal{P}(X) \rightarrow R$
for $i = 1, n$ do
    $c_i = \{x_i\}$ end
$C = \{c_1, \ldots, c_n\}$
$j = n + 1$
while $\mid C \mid > 1$
        $(c_{n_1}, c_{n_2}) = \text{argmax}_{(c_u, c_v) \in C \times C}$ sim$(c_u, c_v)$
        $c_j = c_{n_1} \cup c_{n_2}$
        $C = C \backslash \{c_{n_1}, c_{n_2}\} \cup \{c_j\}$
        $j = j + 1$
```

The k-Means Algorithm

S. P. Lloyd, 1957

Given: a set $X = \{x_1, \ldots, x_n\} \subseteq \mathcal{R}^m$,
a distance measure d on $\mathcal{R}^m$,
a function for computing the mean $\mu : \mathcal{P}(\mathcal{R}^m) \rightarrow \mathcal{R}^m$,

built k clusters so as to satisfy a certain ("stopping") criterion (e.g., maximization of group-average similarity).

Procedure:

```
Select (arbitrarily) $k$ initial centers $f_1, \ldots, f_k$ in $\mathcal{R}^m$;
while the stopping criterion is not satisfied
    for all clusters $c_j$ do $c_j = \{x_i \mid \forall f_l \; d(x_i, f_j) \leq d(x_i, f_l)\}$ end
    for all means $f_j$ do $f_j \leftarrow \mu(c_j)$ end
```

K-Means algorithm revisited (I)

- Se iniţializează în mod arbitrar centroizii $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ şi se ia $C = \{1, \ldots, K\}$.
- Atâta timp cât valoarea criteriului J descreşte în mod strict, repetă:

  Pasul 1:
  Calculează γ astfel:

  $$\gamma_{ij} \leftarrow \begin{cases} 1, & \text{dacă } \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \leq \|\mathbf{x}_i - \boldsymbol{\mu}_{j'}\|^2, \ \forall j' \in C, \\ 0, & \text{în caz contrar.} \end{cases}$$

  În caz de egalitate, alege în mod arbitrar cărui cluster (dintre cele eligibile) să-i aparţină $\mathbf{x}_i$.

  Pasul 2:

  Recalculează $\boldsymbol{\mu}_j$ folosind matricea γ actualizată:
  Pentru fiecare $j \in C$, dacă $\sum_{i=1}^n \gamma_{ij} > 0$, asignează

  $$\boldsymbol{\mu}_j \leftarrow \frac{\sum_{i=1}^n \gamma_{ij} \mathbf{x}_i}{\sum_{i=1}^n \gamma_{ij}}.$$

  Altfel, menţine neschimbat centroidul μ_j.

Algoritmul de clusterizare K-means poate fi văzut [şi reformulat] ca un *algoritm de optimizare*, folosind **metoda *descreşterii pe coordonate***.

Obiectivul este acela de a minimiza o funcţie obiectiv care măsoară (indirect) coeziunea intra-clustere:

$$J(L, \mu) = \sum_{i=1}^n ||x_i - \mu_{l_i}||^2,$$

Algoritmul K-means face *iniţializarea* centroizilor clusterelor μ cu anumite valori, după care procedează astfel:

***Pasul 1*:**
Păstrând μ fixat, găseşte acea asignare L a instanţelor la clustere care minimizează funcţia $J(L, \mu)$;

***Pasul 2*:**
Păstrând asignarea L fixată, găseşte acea valoare pentru μ pentru care se minimizează $J(L, \mu)$.

Criteriul de oprire: Dacă [aceasta nu este prima iteraţie şi] niciuna dintre asignările din lista L nu s-a modificat în raport cu precedenta iteraţie, se trece la pasul următor (Terminare); altfel se repetă de la Pasul 1.

Terminare: Returnează L şi μ.

K-Means algorithm revisited (II)

$$\begin{tikzcd}
\text{t=0} \arrow[d] & & \\
\mu^{(t)} = (\mu_1^{(t)}, \ldots, \mu_K^{(t)}) \arrow[rr] & & \gamma_{ij}^{(t+1)} = \ldots \arrow[ddl] \\
& \boldsymbol{\mathit{min}}\ \mathbf{J(C, \mu)} & \\
& \mu_j^{(t+1)} = \dfrac{\sum \gamma_{ij}^{(t+1)} x_i}{\sum \gamma_{ij}^{(t+1)}} \arrow[uul, "\text{++t}"] &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{t=0} \arrow[d] & & \\
\mu^{(t)} = (\mu_1^{(t)}, \ldots, \mu_K^{(t)}) \arrow[rr] & & C^{(t+1)} = \mathit{argmin}_{C}\ J(C, \mu^{(t)}) \arrow[ddl] \\
& \boldsymbol{\mathit{min}}\ \mathbf{J(C, \mu)} & \\
& \mu^{(t+1)} = \mathit{argmin}_{\mu}\ J(C^{(t+1)}, \mu) \arrow[uul, "\text{++t}"] &
\end{tikzcd}$$

The General EM Problem

Given

- observed data $X = \{x_1, \ldots, x_m\}$ independently generated using the parameterized distributions/hypotheses $h_1, \ldots, h_m$
- unobserved data $Z = \{z_1, \ldots, z_m\}$

determine

$\hat{h}$ that (locally) maximizes $P(X|h)$.

Approach

Start with $h^{(0)}$, an arbitrarily/conveniently chosen value of h.

Repeatedly

1. Use the observed data X and the current hypothesis $h^{(t)}$ to estimate [the probabilities associated to the values of] the unobserved variables Z, and further on compute their expectations, $E[Z]$.
2. The expected values of the unobserved variables Z are used to calculate an improved hypothesis $h^{(t+1)}$, based on maximizing the mean of a log-likelihood function: $E[\ln P(Y|h)|X, h^{(t)}]$, where $Y = \{y_1, \ldots, y_m\}$ is the complete (observed and unobserved) data, i.e. $y_i = (x_i, z_i)$, for $i = 1, \ldots, m$.

The EM algorithmic Schema

$$\begin{tikzcd}
t=0 \arrow[d] & & \\
h^{(t)} \arrow[rr] & & E[Z \mid X, h^{(t)}] \arrow[dd] \\
& \ln P(X|h) & \\
h^{(t+1)} = \underset{h}{\mathit{argmax}}\; E_{P(Z|X;\, h^{(t)})} [\ln P(X,Z|h)] \arrow[uu, "++t"] & & \arrow[ll]
\end{tikzcd}$$

ADMINISTRATIVIA

Who is Liviu Ciortuz?

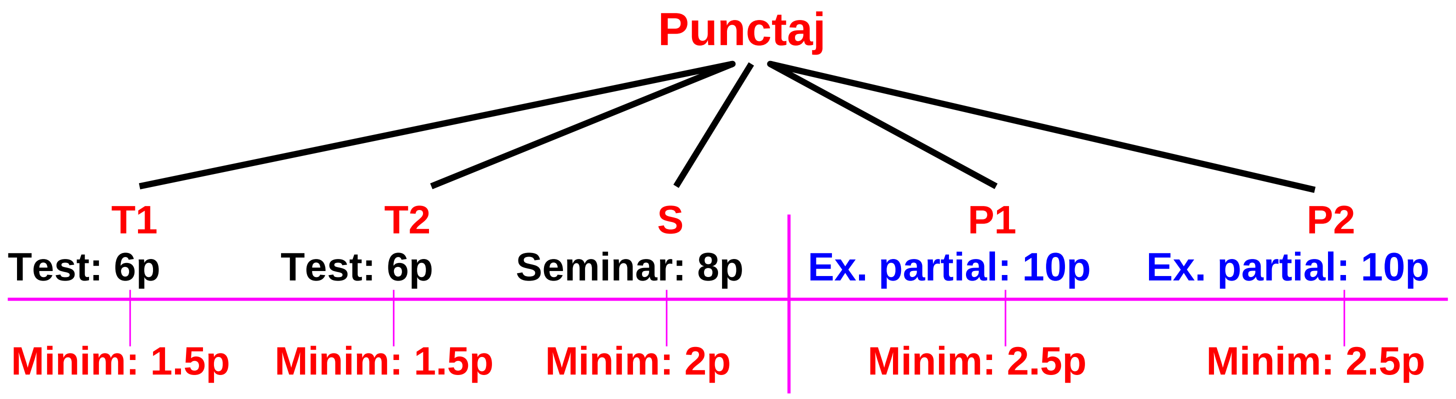
- Diploma (maths and CS) from UAIC, Iaşi, Romania, 1985
  PhD in CS from Université de Lille, France, 1996
- programmer:
  Bacău, Romania (1985-1987)
- full-time researcher:
  Germany (DFKI, Saarbrücken, 1997-2001),
  UK (Univ. of York and Univ. of Aberystwyth, 2001-2003),
  France (INRIA, Rennes, 2012-2013)
- assistant, lecturer, and then associate professor:
  Univ. of Iaşi, Romania (1990-1997, 2003-2012, 2013-today)

Teaching assistants for the ML undergraduate course 2023 (fall semester)

- **Conf. dr. Anca Ignat (. . . Image processing)**
  https://profs.info.uaic.ro/∼ancai/ML/
- **Sebastian Ciobanu (PhD; Amazon)**
  https://sites.google.com/view/seminarml
- **Andi Munteanu**
  (PhD student at UAIC, and research assistent at Univ. of Cambridge)
- **Cristian Simionescu (PhD student; Nexus)**
- **Ramona Albert (PhD student; Amazon)**
- **Ştefan Panţiru (MSc; Mambu)**
- **Corina Dimitriu (MSc student)**

Grading standards for the ML undergraduate course 2023

Obiectiv: Învăţare pe tot parcursul semestrului!



Prezenta la curs: recomandata!
Prezenta la seminar: obligatorie!
Penalizare: 0.2p pentru fiecare absenta de la a doua incolo!

Nota = (10 + T1 + T2 + S + P1 + P2) / 5
Pentru promovare: Nota >= 4.5 <=> T1 + T2 + S + P1 + P2 >= 12.5

REGULI generale pentru cursul de Învăţare automată de la licenţă

Regulile de organizare a cursului de Învăţare Automată (engl., Machine Learning, ML), sem. I, sunt specificate în *fişa disciplinei*
http://profs.info.uaic.ro/∼ciortuz/fisa-disciplinei.pdf

• Bibliografie minimală: vezi slide #8

• Planificarea materiei, pentru fiecare săptămână (curs + seminar):
http://profs.info.uaic.ro/∼ciortuz/what-you-should-know.pdf

• Prezenţa la curs: recomandată!

• **Regula 0:** Prezenţa la seminar: obligatorie!

Pentru fiecare absenţă la seminar, începând de la a doua absenţă încolo, se aplică o penalizare/depunctare de 0.2 puncte. (Vezi formula de notare.)
Regulile se aplică inclusiv studenţilor reînmatriculaţi.

• Săptămânal — marţea, între orele 18–20, în sala C308 — se va ţine un seminar suplimentar, destinat pentru acei studenţi care sunt foarte interesaţi de acest domeniu şi cărora le plac demonstraţiile matematice. (Vedeţi secţiunile “Advanced issues” din documentul http://profs.info.uaic.ro/∼ciortuz/what-you-should-know.pdf.)

REGULI generale pentru cursul de Învăţare automată de la licenţă (cont.)

Regula 1: **Pentru seminarii, nu se admit mutări ale studenţilor de la o grupă la alta, decât în cadrul grupelor care au acelaşi asistent / profesor responsabil de seminar.**

Regula 2: **Nu se fac echivalări de punctaje pentru studenţii care nu au promovat cursul în anii precedenţi.**

Regula 3: **Profesorul responsabil pentru acest curs, Liviu Ciortuz,**
NU va răspunde la email-uri care pun întrebări pentru care raspunsul a fost deja dat
– fie în aceste slide-uri,
– fie pe site-ul Piazza dedicat acestui curs:
https://piazza.com/info.uaic.ro/fall2023/ml2023f/home**,**
– fie la curs.

Recomandare importantă (1) **La fiecare curs şi seminar, studenţii vor avea culegerea de *Exerciţii de învăţare automată* (de L. Ciortuz et al) — vă recomandăm să imprimaţi capitolele *Clasificare bayesiană*, *Învăţare bazată pe memorare*, *Arbori de decizie* şi *Clusterizare* — şi eventual slide-urile indicate în slide-ul următor.**

Recomandare importantă (2) **Consultaţi săptămânal documentul** what-you-should-know.pdf **din pagina de Resurse, de pe site-ul Piazza dedicat acestui curs.**

REGULI generale pentru cursul de Învăţare automată de la licenţă (cont.)

• **Slide-uri de imprimat** (în această ordine şi, de preferat, COLOR):

http://profs.info.uaic.ro/~ciortuz/SLIDES/foundations.pdf

https://profs.info.uaic.ro/~ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.ProbStat.pdf
https://profs.info.uaic.ro/~ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.DT.pdf
https://profs.info.uaic.ro/~ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.Bayes.pdf
https://profs.info.uaic.ro/~ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.IBL.pdf
https://profs.info.uaic.ro/~ciortuz/ML.ex-book/SLIDES/ML.ex-book.SLIDES.Cluster.pdf

(Atenţie: acest set de slide-uri poate fi actualizat pe parcursul semestrului!)

• De imprimat (ALB-NEGRU):

http://profs.info.uaic.ro/~ciortuz/SLIDES/ml0.pdf
http://profs.info.uaic.ro/~ciortuz/SLIDES/ml3.pdf
http://profs.info.uaic.ro/~ciortuz/SLIDES/ml6.pdf
http://profs.info.uaic.ro/~ciortuz/SLIDES/ml8.pdf
http://profs.info.uaic.ro/~ciortuz/SLIDES/cluster.pdf