Supervised learning for text mining

Master Data Mining

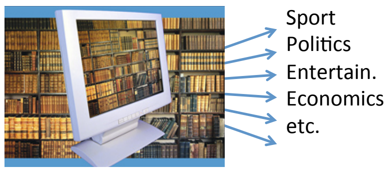
Julien Velcin

Warning

I won't mention deep learning in this course

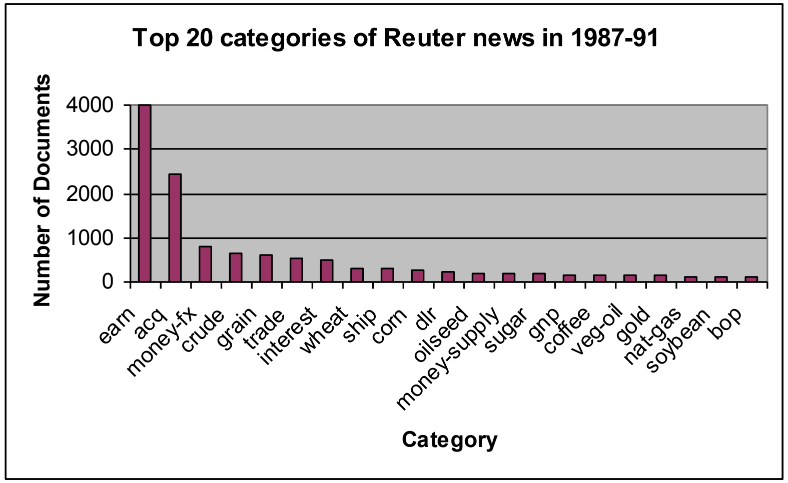
Text classification

- Automatically classify documents into **predefined** classes e.g., digital libraries



- Application areas:
  - Email SPAM filtering
  - Internet directory construction (ex.: Yahoo!)
  - Automatic indexing …

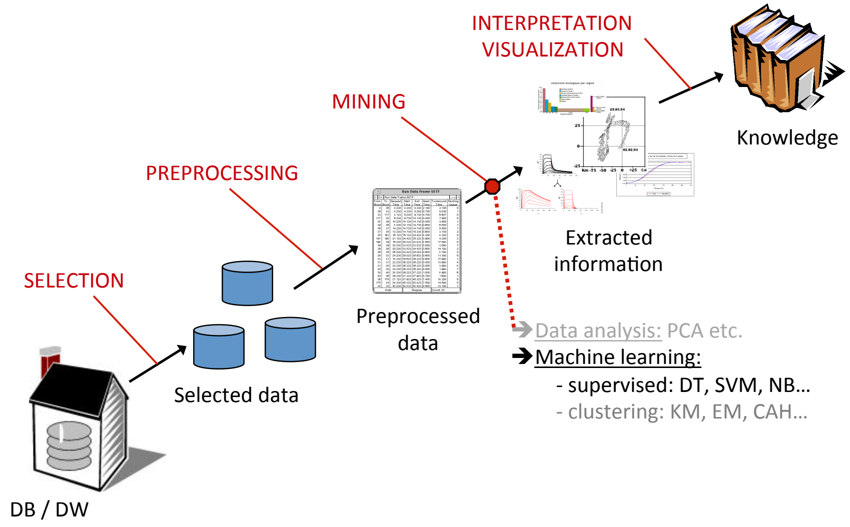
A very classical dataset



Basic ML approach

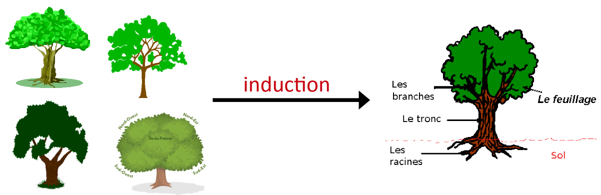
- Prepare a set of **training data**
  - Getting access to the data
  - Need of effective preprocessing
- Create a classifier
  - Apply a ML algorithm: NB, ANN, SVM etc.
- Classify new documents using the **classifier**

ML in data mining



Inductive principle

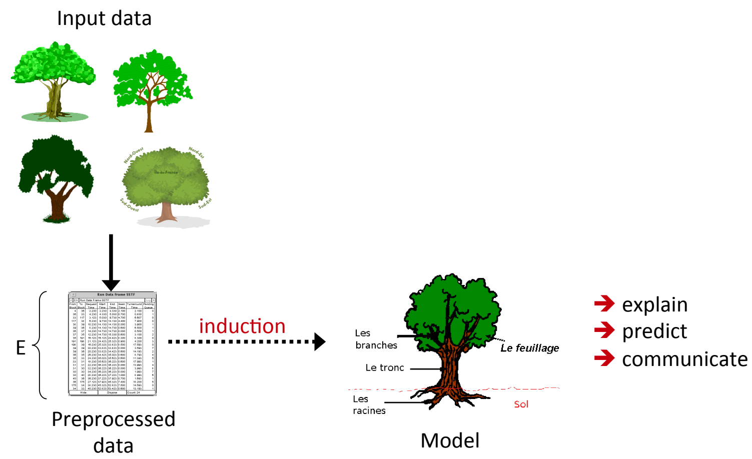
- Aristotle: knowledge comes from the (observable) world



A tree = roots + trunk + branches + leaves

+ relations between: roots and ground, roots and trunk etc.

Induction using a machine



Supervised learning

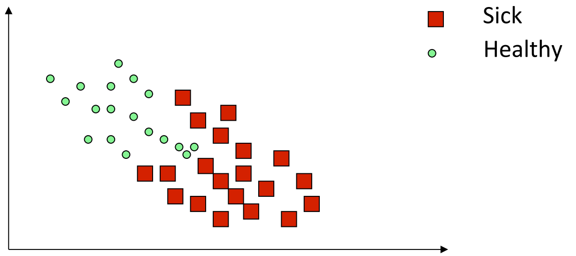
- Inductive learning based on **supervision**
- Given:
  - a representation language L,
  - examples $e_i \in E$ described using L,
  - for each e_i, a class $\Phi(e_i)$ taken from $\{c_1, c_2 \ldots c_p\}$
- The objective is to find:
  - a function (machine) h that relates each description built on L to a class in $\{c_1, c_2 \ldots c_p\}$ ➔ classifieur
  - a function h that relates each description built on L to a real value ➔ régression

A lot of existing algorithms

- K-Nearest Neighbours (KNN)
- Roccio's algorithm
- Decision Trees (DT)
- Artificial Neuronal Networks (ANN)
- Naive Bayes (NB)
- Support Vector Machines (SVM)
- Ensemble methods (bagging, boosting)

etc.

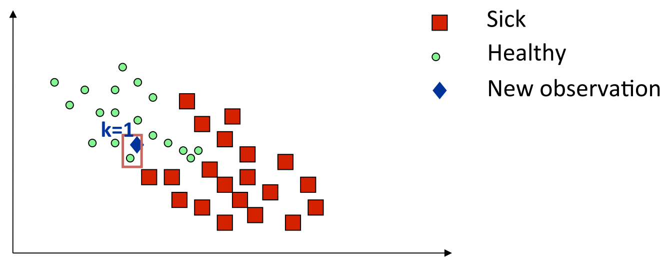
K-Nearest Neighbours



- Using a metric $d(e_i,e_j)$ that calculates the dissimilarity between two individuals e_i and e_j.

K-Nearest Neighbours (cont'd)

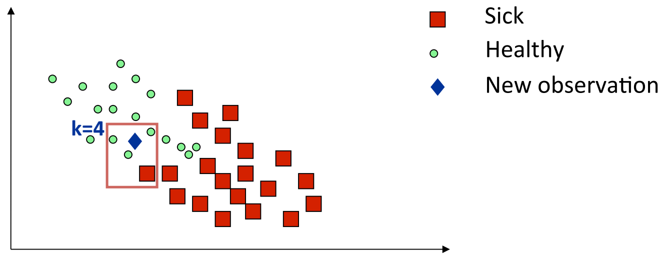
Given a new observation: ♦



Conclusion: we predict that this new person is healthy.

K-Nearest Neighbours (cont'd)

- With k=4:



- **Same conclusion:** we predict that this new person is healthy.

K-Nearest Neighbours (cont'd)

- Pros:
  - Effective
  - Non-parametric
  - Pairwise (local) comparison of documents
- Cons:
  - Classification time is long (usually in $O(n^2)$)
  - Difficult to find an optimal value of k

Rocchio's algorithm

- Build prototype vector for each class
- Prototype vector: average vector over all training document vectors that belong to class c_i

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{\vec{d} \in D_c} \vec{d}$$

- Calculate similarity between test document and each of prototype vectors
- Assign test document to the class with maximum similarity

Rocchio's algorithm (cont'd)

- Pros:
  - Easy to implement
  - Very fast learner
- Cons:
  - Low classification accuracy
  - Only for linear classification boundaries

Need of inductive bias in ML

- Bias = set of assumptions that the learner uses to predict outputs given inputs that it has not encountered [Mitchell, 80]
- Choices for reducing the solution (hypotheses) space; guide of the learning process
- In concept learning: way to favour a generalization rather than another one
- **Absolut need** of bias for learning [Mitchell, 80].
- Examples :
  - Occam's razor,
  - Conditional independance (cf. NB)
  - Maximum margin (cf. SVM)

  etc.

Illustration of the Occam's razor

- Given the sequence: 1, 2, 3, 5 … , a?

Learning as search

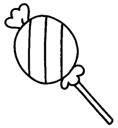
- Choosing a family of concepts (hypotheses)
- Searching for the best hypothesis possible
- To do this, you need fixing biases
- Occam's razor addresses the trade-off:

simplicity / efficiency

Statistical machine learning

- Main concepts:
  - Data (examples, instances, observations)
  - Hypotheses (probabilistic theories on the process of data generation)
- Objective:
  - Explain the observed data
  - Predict new observations

Example [Russel & Norvig, 2003]

- Two flavours for canddies: 
  - *Cherry* 
  - *Lime* 
- Canddies are sold in 5 different bags:

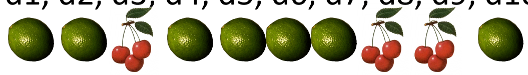
c : 100%	c : 75%	c : 50%	c : 25%	c : 0%
l : 0%	l : 25%	l : 50%	l : 75%	l : 100%
h_1	**h_2**	**h_3**	**h_4**	**h_5**

Example (cont'd)

- Given a new "unknown" bag: 
- H can take 5 values:

  H ∈ {h1, h2, h3, h4, h5}
- Observations: opening the wrappers!

  Ex. : d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, etc.



Baysian learning

- Hypothesis probability given the data:

$$\mathrm{P}(h_i/\mathrm{D}) = \frac{\mathrm{P}(\mathrm{D}/h_i)\mathrm{P}(h_i)}{\mathrm{P}(\mathrm{D})} \propto \mathrm{P}(D/h_i)\mathrm{P}(h_i)$$

- Making predictions:

$$\mathrm{P}(X/\mathrm{D}) = \sum_i \mathrm{P}(X/\mathrm{D},h_i)\mathrm{P}(h_i/\mathrm{D})$$

$$= \sum_i \mathrm{P}(X/h_i)\mathrm{P}(h_i/D)$$

Baysian learning (cont'd)

probability distribution　　　　likelihood

$$\mathrm{P}(X/\mathrm{D}) \propto \sum_i \mathrm{P}(X/h_i)\mathrm{P}(D/h_i)\mathrm{P}(h_i)$$

prior

- How to calculate the likelihood?

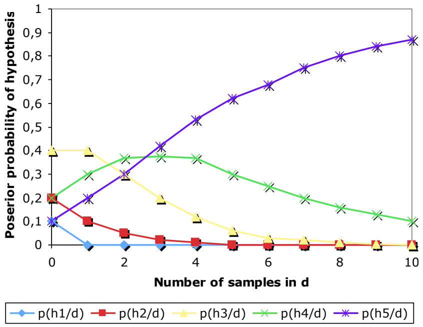
$$\mathrm{P}(D/\mathrm{h}) = \prod_i P(d_i/h)$$

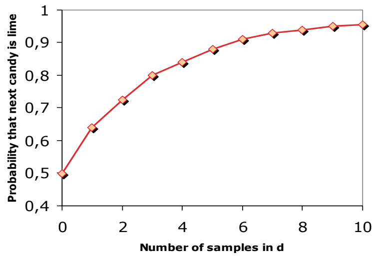
i.i.d. hypothesis (**i**ndependently **i**dentically **d**istributed)

- Example of observed data:

 $\mathrm{P}(D/h_3) = 0{,}5^{10}$

Evolution of learning





Here are the priors: < 0,1 ; 0,2 ; 0,4 ; 0,2 ; 0,1>

Decision in bayesian learning

- Given p(X/D) what decision make?
  - the simpliest way: taking the most likely solution
  - another solution: drawing a random number
- MAP = Maximum A Posteriori
  - here: h_5.

$$\arg\max_{\mathrm{i}} \mathrm{P}(h_i/\mathrm{d})$$

$$\mathrm{P}(X/\mathrm{D}) \approx \mathrm{P}(X/h_{MAP})$$

Role of priors

- $P(h_i)$ very important in Bayesian learning
- Bounding the complexity of models
- Complex models (**hypotheses**) are less likely *a priori*
- **Trade-off** between complexity and expressivity
- If you have no idea on priors: uniform

Link with information theory

- Choosing h_{MAP} for maximizing $\mathrm{P}(\mathrm{D}/h_i)\mathrm{P}(h_i)$ means minimazing:

$$-\log_2 \mathrm{P}(\mathrm{D}/h_i) - \log_2 \mathrm{P}(h_i)$$

- If you split this formula into 2 parts:

$-\log_2 \mathrm{P}(h_i)$ coding hypothis h_i

$-\log_2 \mathrm{P}(\mathrm{D}/h_i)$ additional coding for D

MAP ➔ maximum compression of the data!

Bayesian learning of parametric models

- To begin with: we assume that the data are **complete** (no missing values)
- The objective is to estimate the values of the parameters of a **given model**
- Classical approach: **Maximum-Likelihood Estimation** (MLE) for estimating the model parameters

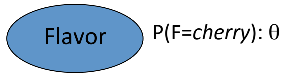
Learning of discrete models

Example: Selling bags containing candies with two different flavours

θ with the cherry flavour

1-θ with the lime flavour

- The problem hypothetis = h_θ.
- Assume uniform priors
- Just one random variable: *Flavour*.



Example (cont'd)

- N observations:
  - o c candies with cherry flavour 
  - o l candies with lime flavour 
- Estimating the likelihood:

$$P(D/h_\theta) = \prod_{j=1}^{N} P(d_j/h_\theta) = \theta^c.(1-\theta)^l$$

- h_{MAP} ➔ taking θ for maximazing this formula
  = maximizing the log-likelihood

Calculating h_θ

- Likelihood: $P(D/h_\theta) = \prod_{j=1}^{N} P(d_j/h_\theta) = \theta^c.(1-\theta)^l$
- Log-likelihood:

$$L(D/h_\theta) = \log P(D/h_\theta) = \sum_{j=1}^{N} \log P(d_j/h_\theta) = c\log\theta + l\log(1-\theta)$$

Which θ maximizes this formula?

$$\frac{\partial L(D/h_\theta)}{\partial\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \quad ➔ \quad \theta = \frac{c}{c+l} = \frac{c}{N}$$

First conclusion

$$\theta = \frac{c}{c+l} = \frac{c}{N}$$

- The **real rate** θ of cherry candies is estimated to the rate of **observed** cherry candies…
- Quite an obvious result! But we have:
  - written a (log)likelihood function,
  - derived this function wrt θ,
  - found the best value for the parameter θ.

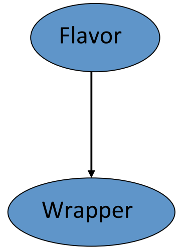
New more complicated example

Adding wrappers:
red and green

Model with 3 parameters:

$\theta, \theta_1, \theta_2$

P(*F*=*cherry*) : θ



F	P(*W*=red/*F*)
cherry	θ_1
lime	θ_2

What is the likelihood of a cherry candy in a green wrapper?

$$\mathrm{P}(F=c, W=g / h_{\theta,\theta_1,\theta_2})$$
$$= \mathrm{P}(F=c / h_{\theta,\theta_1,\theta_2})\mathrm{P}(W=g / F=c, h_{\theta,\theta_1,\theta_2}) = \theta.(1-\theta_1)$$

Estimating the likelihood

- Unwrapping N candies:
  - o c with cherry flavour, l with lime flavour,
  - o cherry candies: r_c red wrappers, g_c green wrappers,
  - o lime candies: r_l red wrappers, g_l green wrappers.

$$\mathrm{P}(\mathrm{D}/h_{\theta,\theta_1,\theta_2}) = \theta^c.(1-\theta)^l.\theta_1^{r_c}.(1-\theta_1)^{g_c}.\theta_2^{r_l}.(1-\theta_2)^{g_l}$$

$$\mathrm{L} = [\mathrm{c}.\log\theta + l.\log(1-\theta)] + [r_c \log\theta_1 + g_c \log(1-\theta_1)]$$
$$+ [r_l \log\theta_2 + g_l \log(1-\theta_2)]$$

Calculating $h_{\theta,\theta1,\theta2}$

- Derivations:

$$\frac{\partial \mathrm{L}}{\partial\theta} = \frac{c}{\theta} - \frac{l}{1-\theta} = 0 \rightarrow \theta = \frac{c}{c+l}$$

$$\frac{\partial \mathrm{L}}{\partial\theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \rightarrow \theta_1 = \frac{r_c}{r_c+g_c}$$

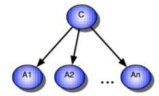
$$\frac{\partial \mathrm{L}}{\partial\theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1-\theta_2} = 0 \rightarrow \theta_2 = \frac{r_l}{r_l+g_l}$$

- Same conclusion…

Preliminary conclusions

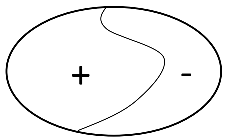
- Results fit our expectations
- Using complete data and h_{MAP}, Bayesian learning split the problem into **subproblems** (one for each model parameter)
- **Mind** the "zero probability" situation (especially for small datasets)

Naive Bayes



- Naive Bayes (NB) is a classical classifier in ML
- Based on the **Bayes' Theorem**
- Strong independence assumptions
- Easy and quick to implement
- Good results in practice

Basics on NB



- Example: e = tuple (x_1, x_2…, x_m) + class
- Case of binary classification: $c \in \{+,-\}$
- Learning set:

$$E = \{(e_1,+),(e_2,+),(e_3,-)\ldots, (e_n,+)\}$$

- We are looking for a function f_B so that:

$$f_B(e) = \frac{p(C=+/e)}{p(C=-/e)}$$

- If $f_B \geq 1$ then h(e)=+ else h(e)=-

Calculating f_B

- How to calculate p(c/e)?
- Bayes' rule:

$$p(c/e) = \frac{p(e/c) \times p(c)}{p(e)}$$

- And now… p(e/c) ?

$$p(e/c) = p(x_1, x_2, x_3 \ldots, x_m / c)$$

- What else?

conditional independence

We state the following fundamental hypothesis:

Hypothesis: We assume that all the attributes are **independent** given the associated class.

So:

$$\mathrm{p}(x_1 / x_2, x_3 ..., x_m, c) = \mathrm{p}(x_1 / c)$$

$$\mathrm{p}(e / c) = p(x_1, x_2, x_3 ..., x_m / c) = \prod_{i=1}^{m} \mathrm{p}(x_i / c)$$

The Naive Bayes classifier (NB)

- With this hypothesis, $f_B \rightarrow f_{NB}$:

$$f_{NB}(e) = \frac{\mathrm{p}(C=+)}{\mathrm{p}(C=-)} \prod_{i=1}^{m} \frac{\mathrm{p}(x_i / C=+)}{\mathrm{p}(x_i / C=-)}$$

with:

- o p(*C*=+): proportion of positive instances,
- o p(*C*=-): proportion of negative instances,
- o p(x_i/*C*=+): proportion of positive instances with X=x_i
- o p(x_i/*C*=-): proportion of negative instances with X=x_i

Illustration (Quinlan,83)

Attributs	**Pif**	**Temp**	**Humid**	**Vent**	**Golf**
Valeurs	soleil, couvert, pluie	chaud, bon, frais	normale, haute	vrai, faux	jouer, ne_pas_jouer

1	soleil	chaud	haute	faux	ne_pas…
2	soleil	chaud	haute	vrai	ne_pas…
3	couvert	chaud	haute	faux	jouer
4	pluie	bon	haute	faux	jouer
5	pluie	frais	normale	faux	jouer
6	pluie	frais	normale	vrai	ne_pas…
7	couvert	frais	normale	vrai	jouer
8	soleil	bon	haute	faux	ne_pas…
9	soleil	frais	normale	faux	jouer
10	pluie	bon	normale	faux	jouer
11	soleil	bon	normale	vrai	jouer
12	couvert	bon	haute	vrai	jouer
13	couvert	chaud	normale	faux	jouer
14	pluie	bon	haute	vrai	ne_pas…

class

jouer = classe + (9 exemples sur 14)

1	soleil	chaud	haute	faux	ne_pas…
2	soleil	chaud	haute	vrai	ne_pas…
3	couvert	chaud	haute	faux	jouer
4	pluie	bon	haute	faux	jouer
5	pluie	frais	normale	faux	jouer
6	pluie	frais	normale	vrai	ne_pas…
7	couvert	frais	normale	vrai	jouer
8	soleil	bon	haute	faux	ne_pas…
9	soleil	frais	normale	faux	jouer
10	pluie	bon	normale	faux	jouer
11	soleil	bon	normale	vrai	jouer
12	couvert	bon	haute	vrai	jouer
13	couvert	chaud	normale	faux	jouer
14	pluie	bon	haute	vrai	ne_pas…

Attributs	**Pif**	**Temp**	**Humid**	**Vent**	**Golf**
Valeurs	Soleil **(2)**, couvert **(4)**, pluie **(3)**	chaud **(2)**, bon **(4)**, frais **(3)**	normale **(6)**, haute **(3)**	vrai **(3)**, faux **(6)**	**jouer**, ne_pas_jouer

ne_pas_jouer = classe - (5 exemples sur 14)

1	soleil	chaud	haute	faux	ne_pas…
2	soleil	chaud	haute	vrai	ne_pas…
3	couvert	chaud	haute	faux	jouer
4	pluie	bon	haute	faux	jouer
5	pluie	frais	normale	faux	jouer
6	pluie	frais	normale	vrai	ne_pas…
7	couvert	frais	normale	vrai	jouer
8	soleil	bon	haute	faux	ne_pas…
9	soleil	frais	normale	faux	jouer
10	pluie	bon	normale	faux	jouer
11	soleil	bon	normale	vrai	jouer
12	couvert	bon	haute	vrai	jouer
13	couvert	chaud	normale	faux	jouer
14	pluie	bon	haute	vrai	ne_pas…

Attributs	**Pif**	**Temp**	**Humid**	**Vent**	**Golf**
Valeurs	Soleil **(3)**, couvert **(0)**, pluie **(2)**	chaud **(2)**, bon **(2)**, frais **(1)**	normale **(1)**, haute **(4)**	vrai **(3)**, faux **(2)**	jouer, **nepas_jouer**

Calculating the probabilities

p(C=+) = p(jouer) = 9/14 ≈ 0,64

p(C=-) = p(ne_pas_jouer) = 5/14 ≈ 0,36

Attributs	**Pif**	**Temp**	**Humid**	**Vent**	**Golf**
Valeurs	Soleil **(2)**, couvert **(4)**, pluie **(3)**	chaud **(2)**, bon **(4)**, frais **(3)**	normale **(6)**, haute **(3)**	vrai **(3)**, faux **(6)**	**jouer**
p(x_i / C=+)	**0,22 ; 0,45 ; 0,33**	**0,22 ; 0,45 ; 0,33**	**0,67 ; 0,33**	**0,33 ; 0,67**	

Attributs	**Pif**	**Temp**	**Humid**	**Vent**	**Golf**
Valeurs	Soleil **(3)**, couvert **(0)**, pluie **(2)**	chaud **(2)**, bon **(2)**, frais **(1)**	normale **(1)**, haute **(4)**	vrai **(3)**, faux **(2)**	**nepas_jouer**
p(x_i / C=-)	**0,6 ; 0 ; 0,4**	**0,4 ; 0,4 ; 0,2**	**0,2 ; 0,8**	**0,6 ; 0,4**	

Predicting the class of a new observation

- Bob would like to know whether he can go to play golf. He observes that the weather is **sunny**, the temperature is **high**, the humidity is **normal** and there is **no wind**.

  o = {(Pif=soleil), ($Temp$=bon), ($Humid$=normal), ($Vent$=faux)}

$$f_{NB}(o) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{m} \frac{p(x_i / C=+)}{p(x_i / C=-)}$$

o = {(P=s), (T=b), (H=n), (V=f)}

Predicting the class of a new observation (cont'd)

$$f_{NB}(o) = \frac{p(C=+)}{p(C=-)} \prod_{i=1}^{m} \frac{p(x_i / C=+)}{p(x_i / C=-)} = \frac{9}{14}\,\frac{14}{5} \prod_{i=1}^{m} \frac{p(x_i / C=+)}{p(x_i / C=-)}$$

$$f_{NB}(o) = \frac{9}{5}\left[\frac{p(P=s/+)}{p(P=s/-)} \cdot \frac{p(T=b/+)}{p(T=b/-)} \cdot \frac{p(H=n/+)}{p(H=n/-)} \cdot \frac{p(V=f/+)}{p(V=f/-)}\right]$$

$$f_{NB}(o) \approx \frac{9}{5}\left[\frac{0{,}22}{0{,}6} \cdot \frac{0{,}45}{0{,}4} \cdot \frac{0{,}67}{0{,}2} \cdot \frac{0{,}67}{0{,}4}\right]$$

$$f_{NB}(o) \approx 1{,}8 \times [0{,}37 \times 1{,}13 \times 3{,}35 \times 1{,}68] \approx 4{,}24 \geq 1$$

Conclusion: Bob can go to play golf!

Pros of NB

- Simplicity of the theoretical background
- Memory complexity = probability tables
- Calculus are really fast: mainly x
- Often a good baseline for experiments
- Good results for real datasets

Cons of NB

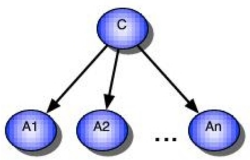
- Strong hypotheses (e.g., the conditional independence)
- "Zero-probability" issue

  E.g., test with o = {(*P*=c), (*T*=ch), (*H*=h), (*V*=v)}
- Theoretical results not really well known

Link with Bayesian networks

- Very simple form of a Bayesian network

$$\forall X_{\mathrm{i}}, \mathrm{p}(X_i = x_{ij} / X_1, X_2 ..., X_m, C) = \mathrm{p}(X_i = x_{ij} / C)$$



Managing continuous variables

- Basic NB deal only with discrete variables
- Possible solutions for continuous variables:
  - o discretizing the variables,
  - o adding an **hypothesis** on the data distribution (e.g., using a Gaussian distribution)

$$\mathrm{P}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$