Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

CAS Practical Machine Learning
Introduction

# Supervised Learning

Prof. Dr. Jürgen Vogel (juergen.vogel@bfh.ch)

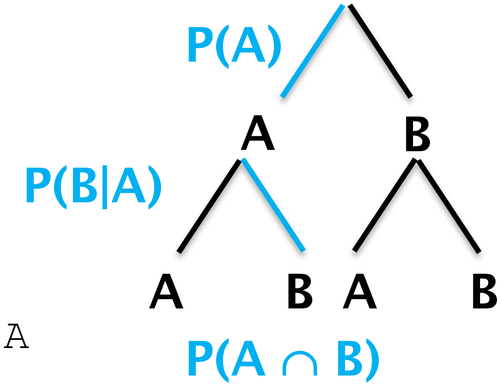
# Supervised Learning

an algorithm learns from experience `E` to solve some tasks `T` with performance `P` if `P` improves with `E`

Supervised Learning

- tasks `T` that are solved: mapping a sample (based on its features) to
  - some output class
    - classification
      - e.g., Naïve Bayes (NB)
  - some data structure (typically a tree or network)
    - structured prediction
      - e.g., Bayesian networks
  - some ranking (in relation to other samples)
    - learning to rank
      - e.g., RankNet
- the ML algorithm infers the model from sample data `E` for which the task `T` has been solved with optimal performance `P`
- the algorithm learns directly from the given sample data

# Bayes' Theorem




- mathematical law about conditional probabilities
- given two events $A$ and $B$, then the conditional probability $P(B|A)$ relates to the probability that event $B$ occurs after $A$ has occurred
  - applies, e.g., when we are blindly drawing samples from a bag containing red and black balls without returning the balls
    - assume we start with 2 red and 2 black balls
      - then $P(red) = 2/4 = P(black)$ for the first draw
    - if we draw a red ball first, then we know upfront for the 2nd draw
      - $P(red|red) = 1/3$ and $P(black|red) = 2/3$
- probability that two conditionally-related events $P(A)$ and $P(B|A)$ occur one after another: $P(A \cap B) = P(A) * P(B|A)$
  - e.g., probability to first draw red and then red again: $P(red \cap red) = P(red) * P(red|red) = 2/4 * 1/3 = 1/6$
- $P(A|B)$, $P(B|A)$ may be calculated as
  - $P(A|B) = P(A \cap B) / P(B)$ and $P(B|A) = P(A \cap B) / P(A)$ (for $P(B) \neq 0$ and $P(A) \neq 0$)
  - in the drawing example, $P(red|red) = 2/4 * 1/3 / 2/4 = 1/3$
- from this we can derive Bayes' theorem
  - $P(A|B) = P(B|A) P(A) / P(B)$

# Using Bayes' Theorem for Classification

- our events are: `A` = class c, `B` = sample x
- we calculate `P(c|x)`
  - the probability that a given sample belongs into a certain class
  - based on certain features of the sample
- given `x`, our classfier thus
  1. calculates `P(c|x)` for all `c`
  2. selects `c` with the highest `P(c|x)`

# Using Bayes' Theorem for Classification

Calculating `P(c|x) = P(x|c) P(c) / P(x)`

- on the basis of a representative (and large) training set
  - i.e., all probabilities are estimated as relative frequencies
- `P(x|c)` probability of a sample `x` given the class `c`

  $= P(f_1, f_2, ..., f_n|c)$ where $f_i$ are the sample's features
  - we assume that $f_i$ are independent from each other given a class `c`
    - thus we can calculate $P(f_1,f_2,\ldots,f_n|c)=P(f_1|c)*P(f_2|c)*\ldots*P(f_n|c)$
    - 3 different NB variants for calculating $P(f_i|c)$
      - Gaussian: $f_i$ are continuous and $P(f_i|c)$ are normally distributed
      - multinomial: $P(f_i|c)$ = # of times $f_i$ occurs in c / # of times $f_i$ occurs overall
      - Bernoulli: binary features
  - in general this independence assumption is wrong
    - within `c` certain features are often correlated
    - thus the name: **naive bayes** classifier (NB)
    - but the NB classifier performs well in practice despite this "naive" assumption
- `P(c)` probability of a class `c`

  `= # of samples within c / # of all samples`
- `P(x)` probability of a sample `x`

  `= 1 / # of samples`
  - is equal for all possible `P(c|x)` and thus irrelevant for the actual classification decision (just a scaling factor)

# Example: Text Classifier with Words as Features

| | Sample | Features | Class |
|---|---|---|---|
| Train | 1 | Chinese Beijing Chinese | cn |
| | 2 | Chinese Chinese Shanghai | cn |
| | 3 | Chinese Macao | cn |
| | 4 | Tokyo Japan Chinese | jp |
| Test | 5 | Chinese Chinese Chinese Tokyo Japan | ? |

- P(c) = # of training samples within c / # of all training samples
  - P(cn) = 3/4
  - P(jp) = 1/4
- $P(x|c) = P(f_1|c) * P(f_2|c) * \ldots * P(f_n|c)$
  - # of words in cn: 8; in jp: 3
  - P(Chinese|cn) = (5+1) / 8 = 6/8
  - P(Tokyo|cn) = (0+1) / 8 = 1/8
  - P(Japan|cn) = (0+1) / 8 = 1/8
  - P(Chinese|jp) = (1+1) / 3 = 2/3
  - P(Tokyo|jp) = (1+1) / 3 = 2/3
  - P(Japan|jp) = (1+1) / 3 = 2/3
- P(c|x) = P(x|c) P(c) / P(x)
  - $P(cn|x_5)$ = 6/8 * 6/8 * 6/8 * 1/8 * 1/8 * 3/4 = 0.00494
  - $P(jp|x_5)$ = 2/3 * 2/3 * 2/3 * 2/3 * 2/3 * 1/4 = 0.03

**Notes**:

- P's for Beijing, Shanghai, and Macao are not listed
- in order to prevent that $P(f_i|c)$ is 0, we always add 1
- this variant of NB is called **multinomial** because we add up the occurrences
- if we only note the occurrence with 0 or 1, we have Bernoulli NB

# Underflow Prevention

- Multiplying many of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since `log(xy) = log(x) + log(y)`, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

# Summary Naive Bayes Classifier

- Classify based on prior weight of class and conditional parameter for what each word says:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \left[ \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j) \right]$$

- Training is done by counting and dividing:

$$P(c_j) \leftarrow \frac{N_{c_j}}{N} \qquad P(x_k \mid c_j) \leftarrow \frac{T_{c_j x_k} + \alpha}{\sum_{x_i \in V} [T_{c_j x_i} + \alpha]}$$

# Naive Bayes is not so Naive

Advantages

- Very fast learning and testing
  - basically just count feature occurrences
- Low storage requirements
- Optimal if the independence assumptions hold
- Very good in domains with many equally important features
- More robust to irrelevant features than many other learning methods
  - Irrelevant features cancel each other without affecting results
- A good dependable baseline classifier

Disadvantages

- Often not the best classifier