

# Bayesian learning

Maria Simi, 2008/2009

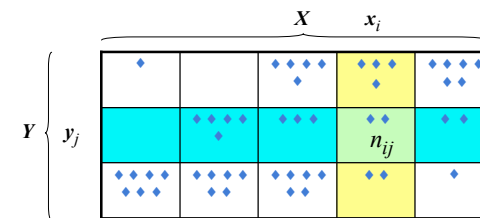
## Bayesian learning

- Relevance to machine learning:
  - The naïve Bayes classifiers are one of the most practical approaches in a number of problems, e.g. text classification.
  - Bayesian methods provide an explanation of learning algorithms that do not explicitly manipulate probabilities
- Features
  - Each training example can incrementally increase or decrease the estimated probability of an hypothesis
  - Prior knowledge can be accommodated: by prior probabilities and by assumptions on probability distribution
  - Hypothesis can be weighted by probabilities and combined
  - Often computationally intractable but
    - Useful as a standard of optimal decision making
    - Practical under simplifying assumptions

## Plan

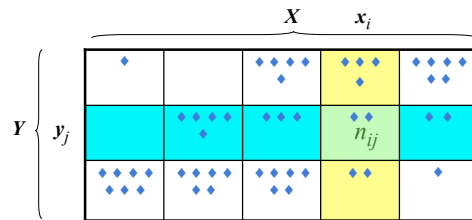
- A graphical reminder of probability
- Bayes Theorem
- Concept learning revisited
- Decision trees and minimum description length
- Optimal Bayes classifier
- The naïve assumption of independence
- Zero Counts and Smoothing

## Probability



- Two random variables  $X = \{x_1, x_2, x_3, x_4, x_5\}$  and  $Y = \{y_1, y_2, y_3\}$
- ♦ trials ( $N = 50$ );  $n_{ij}$  number of trials in  $(i, j)$ , with  $X = x_i$  and  $Y = y_j$
- $c_i$  trials in column  $i$ ;  $r_j$  trials in row  $j$
- $P(X = x_i, Y = y_j) = n_{i,j}/N = 2/50$       joint probability
- $P(X = x_i) = c_i/N = 8/50$       [marginal] probability of  $X$
- $P(Y = y_j) = r_j/N = 12/50$       [marginal] probability of  $Y$
- $P(X) = \sum_y P(X, Y)$       Sum rule

## Conditional Probability



- $P(Y = y_j | X = x_i) = n_{ij} / c_i = 2/8$  *conditional probability*
- $P(Y = y_j, X = x_i) = n_{ij} / N = n_{ij} / c_i \times c_i / N = P(Y = y_j | X = x_i) P(X = x_i)$  *product rule*
- $P(X, Y) = P(Y | X) P(X)$  *product rule*
- $P(X, Y) = P(X | Y) P(Y)$  *product rule*
- $P(Y | X) P(X) = P(X | Y) P(Y)$

## Bayes theorem

$$P(Y | X) = \frac{P(X | Y) P(Y)}{P(X)} \quad \text{Bayes Theorem}$$

$$P(X) = \sum_y P(X | Y) P(Y) \quad \text{Sum rule + Product rule}$$

$$P(Y | X) = \frac{P(X | Y) P(Y)}{\sum_y P(X | Y) P(Y)} \quad \text{Bayes Theorem}$$

Nel nostro caso:

$$P(Y = y_j | X = x_i) = P(X = x_i | Y = y_j) P(Y = y_j) / P(X = x_i) = (2/12 \times 12/50) / (8/50) = 1/25 \times 25/4 = 1/4$$

## Bayes theorem and concept learning

- Given:
  - $H$  hypothesis space, where  $h$  is  $c: X \rightarrow \{0, 1\}$
  - $D$  training data  $\langle \langle x_1, d_1 \rangle, \dots, \langle x_N, d_N \rangle \rangle$

$$P(h | D) = \frac{P(D | h) P(h)}{P(D)}$$

- Compute the posterior probability of each hypothesis  $h$ , using Bayes theorem
- Output the most probable,  $h_{MAP}$

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h | D) = \operatorname{argmax}_{h \in H} P(D | h) P(h)$$

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D | h) \quad \text{when assuming a uniform distribution}$$

## The assumptions for concept learning

- $P(h)$  and  $P(D|h)$  under the assumptions of concept learning:

- $D$  is noise free
- The concept to be learned is in  $H$
- Every hypothesis has *uniform* probability

$$P(h) = \frac{1}{|H|} \quad P(D|h) = 1 \text{ if } h \text{ consistent with } D$$

$$P(D|h) = 0 \text{ otherwise}$$

$$P(h|D) = \begin{cases} \frac{1}{|S_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$$

## Minimum description length principle

- Inductive bias of Decision tree learning (Occam's razor): a Bayesian perspective

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(D | h) P(h)$$

$$= \operatorname{argmax}_{h \in H} \log_2 P(D | h) + \log_2 P(h)$$

$$= \operatorname{argmin}_{h \in H} -\log_2 P(D | h) - \log_2 P(h)$$

$L_{C_1}(h) = -\log_2 P(h)$ : optimal encoding for hypothesis  $h$

$L_{C_2}(D | h) = -\log_2 P(D | h)$ : optimal encoding for training data  $D$ , knowing hypothesis  $h$

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D | h) \quad \text{Minimum Description Length}$$

## Minimum Description Length: interpretation

- Given encodings  $C_1$  and  $C_2$  choose the hypothesis that minimizes  $L_{C_1}(h) + L_{C_2}(D | h)$
- $C_1$  encoding of decision trees
- $C_2$  needs only to encode misclassified examples
- So it may prefer a shorter hypothesis with a few errors or a longer one that perfectly classifies the training data
- MDL can be used choose the best size for DT's
- MDL produced results comparable to the other pruning methods
- Theoretical relevance: *if* the encodings agree with probabilities we have a MAP hypothesis, but this is difficult to assess

## Bayes classifier

- The relevant question is: "What is the most likely *classification* of the new instance given the training data?"
- Not the same as finding and applying the MAP hypothesis. Assume:
  - $H = \{h_1, h_2, h_3\}$
  - $P(h_1|D) = 0,4$ ;  $P(h_2|D) = 0,3$ ;  $P(h_3|D) = 0,3$
  - $x$  classified positive by  $h_1$ ; negative by  $h_2$  and  $h_3$
- Classification according to *MAP* is *positive*
- Most likely classification is *negative*

## Bayes optimal classifier

- The most probable classification of the new instance is obtained by combining the prediction of all hypotheses weighted by their posterior probability
- $C = \{c_1, \dots, c_k\}$  possible classifications labels

$$P(c_j|D) = \sum_{h_i \in H} P(c_j | h_i) P(h_i | D)$$

is the probability of the  $c_j$  classification given  $D$

- Bayes *optimal* classification of a new instance:

$$c_{MAP} = \operatorname{argmax}_{c_j \in V} \sum_{h_i \in H} P(c_j | h_i) P(h_i | D)$$

## Example

$$P(h_1|D) = 0,4; P(-|h_1) = 0; P(+|h_1) = 1$$

$$P(h_2|D) = 0,3; P(-|h_2) = 1; P(+|h_2) = 0$$

$$P(h_3|D) = 0,3; P(-|h_3) = 1; P(+|h_3) = 0$$

$$P(+|D) = \sum_{h_i=H} P(+|h_i) P(h_i|D) = 0,4$$

$$P(-|D) = \sum_{h_i=H} P(-|h_i) P(h_i|D) = 0,6$$

$$\operatorname{argmax}_{v_j \in \{+, -\}} \sum_{h_i=H} P(v_j|h_i) P(h_i|D) = -$$

- Note: the prediction can correspond to a hypothesis non contained in  $H$ , a sort of linear combination of the hypotheses in  $H$
- It can be very costly to compute.

## Naïve Bayes classifier

- A more practical approach is the naïve Bayes classifier
- We assume instances  $x$  described as conjunction of attributes  $a_1 a_2 \dots a_n$ . The target function takes values in  $C$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(c_j | a_1 a_2 \dots a_n)$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} \frac{P(a_1 a_2 \dots a_n | c_j) P(c_j)}{P(a_1, a_2, \dots, a_n)}$$

$$c_{MAP} = \operatorname{argmax}_{c_j \in C} P(a_1 a_2 \dots a_n | c_j) P(c_j)$$

- How to estimate  $P(c_j)$  and  $P(a_1 a_2 \dots a_n | c_j)$  over the training examples?

## Naïve Bayes classifier

- Estimating all the  $P(a_1 a_2 \dots a_n | c_j)$  may be problematic unless we have a large set of training data
- The naïve Bayes classifier is based in the simplifying assumption that the attribute values are independent

$$P(a_1 a_2 \dots a_n | c_j) = P(a_1 | c_j) \times \dots \times P(a_n | c_j) \\ = \prod_i P(a_i | c_j)$$

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(a_i | c_j) \quad \text{by the simplifying independence assumption}$$

- How to compute  $P(c_j)$  and  $P(a_i | c_j)$ ?

## Probability estimates: maximum likelihood

- The simplest way to estimate  $P(c_j)$  and  $P(a_i | c_j)$  is by computing frequencies on the training set:

$$P(c_j) = |c_j|/N$$

$$P(a_i | c_j) = |a_i, c_j|/|c_j|$$

the contribution of a feature  $a_i$  toward the label  $c_j$

- In general it is a good estimate when there are many examples.

## Applying Naive Bayes to Play Tennis example

- Let's classify the new instance

<Outlook=Sunny, Temp=Cool, Humidity=High, Wind=Strong>

given the same training  $D$

$$c_{NB} = \underset{c_j \in \{yes, no\}}{\operatorname{argmax}} P(c_j) P(\text{Outlook}=\text{Sunny} | c_j) P(\text{Temp}=\text{Cool} | c_j) P(\text{Humidity}=\text{High} | c_j) P(\text{Wind}=\text{Strong} | c_j)$$

$$P(\text{PlayTennis}=\text{yes}) = 9/14=0,64 \quad P(\text{PlayTennis}=\text{no}) = 5/14=0,36$$

...

$$P(\text{Wind}=\text{Strong} | \text{PlayTennis}=\text{yes}) = 3/9 = 0,33$$

$$P(\text{Wind}=\text{Strong} | \text{PlayTennis}=\text{no}) = 3/5 = 0,60$$

$$P(\text{yes}) P(\text{Sunny} | \text{yes}) P(\text{Cool} | \text{yes}) P(\text{High} | \text{yes}) P(\text{Strong} | \text{yes}) = 0,0053$$

$$P(\text{no}) P(\text{Sunny} | \text{no}) P(\text{Cool} | \text{no}) P(\text{High} | \text{no}) P(\text{Strong} | \text{no}) = 0,0206$$

The classification is *no* with probability  $0,0206 / (0,0206 + 0,0053) = 0,795$

## Problem with probabilities estimates

- Estimating probabilities by  $n_c / n$  (occurrences of an event over the total number of opportunities) is problematic when  $n_c$  is very small or zero.
- Example suppose

$$P(\text{Wind}=\text{Strong} | \text{PlayTennis}=\text{no}) = 0$$

because we have only few instances with *PlayTennis=no* and none of them has *Wind=Strong*; this underestimate of probability will dominate in classifying new instances with *Wind=Strong*, regardless of the other attributes.

## Smoothing and alternative estimation

- Smoothing* techniques are used to mitigate this effect; one of these techniques is the *m*-estimate
- m*-estimate of  $p$  of a probabilistic event

$\text{max-likelihood} = n/N$	relative frequency of the event
$\text{m-estimate} = \frac{n+mp}{N+m}$	$p$ prior probability, e.g. $1/k$ with $k$ the number of possible outcomes $m$ equivalent sample size

when  $p=1/k$  we pretend to add  $m$  virtual samples with a uniform distribution ( $p$  may be a different distribution)

## The *naïve* assumption of independence

- In some cases it is unreasonable to assume that all features are independent
- What happens when we use the naive Bayes classifier with features that are not independent?
- Risk of "*double-counting*" the effect of highly correlated features.  
Ex. "Ends with 'a'" and "Ends with vowel" in a task of classifying first names according to gender

## References

- Machine Learning, Tom Mitchell, Mc Graw-Hill International Editions, 1997 (Cap 5).