Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Outline

- Naïve Bayes algorithm
- Bayesian networks

Bayes theorem

Probabilistic approach to classification $Y = \{y_{1,2}, \ldots, y_K\}$

$$y^\star = \text{argmax}_{y_k \in Y} \Pr(y_k | x_1, \ldots, x_m) \tag{1}$$

Bayes theorem

$$\text{posterior probability} = \frac{\text{prior probability} \times \text{likelihood}}{\text{marginal likelihood}} \tag{2}$$

$$\Pr(Y \mid A_1, \ldots, A_m) = \frac{\Pr(Y) \cdot \Pr(A_1, \ldots, A_m \mid Y)}{\Pr(A_1, \ldots, A_m)}$$

Then

$$y^\star = \text{argmax}_{y_k \in Y} \frac{\Pr(y_k) \cdot \Pr(x_1, \ldots, x_m | y_k)}{\Pr(x_1, \ldots, x_m)} \tag{3}$$

Conditional independence

Let X, Y and Z be three descrete random variables. We say that X is *conditionally independent* of Y given Z if

$\forall x_i, y_j, z_k, x_i \in \mathit{Values}(X), y_j \in \mathit{Values}(Y), z_k \in \mathit{Values}(Z)$:

$$\Pr(X = x_i | Y = y_j, Z = z_k) = \Pr(X = x_i | Z = z_k) \tag{4}$$

I.e., $\mathrm{P}(X|Y, Z) = \mathrm{P}(X|Z)$.

Conditional independence

Do we enjoy our favorite water sport on this day? (Credit: T. Mitchel, 1997)

Sky	AirTemp	Humidity	Wind	EnjoySport
sunny	warm	normal	strong	No
sunny	warm	high	strong	Yes
rainy	cold	high	strong	No
sunny	warm	high	strong	Yes

Conditional independence of features given *EnjoySport*: presence of one particular feature value does not affect the other features' values given *EnjoySport*, e.g., if the temperature is hot, it does not necessarily mean that the humidity is high and the features have an equal effect on the outcome

Conditional independence

If we work with two features A_1, A_2 and we assume that they are conditionally independent given the target class Y, then

$$\Pr(A_1, A_2|Y) \overset{\text{product rule}}{=} \Pr(A_1|A_2, Y) \cdot \Pr(A_2|Y) \overset{\text{c. i. assumption}}{=} \Pr(A_1|Y) \cdot \Pr(A_2|Y)$$

Note: Product rule (a.k.a. Chain rule)

$$\Pr(A_m, \ldots, A_1) = \Pr(A_m|A_{m-1}, \ldots, A_1) \cdot \Pr(A_{m-1}, \ldots, A_1)$$

Naïve Bayes classifier

$$y^\star = \text{argmax}_{y_k \in Y} \Pr(y_k | x_1, \ldots, x_m) = \text{argmax}_{y_k \in Y} \frac{\Pr(y_k) \cdot \Pr(x_1, \ldots, x_m | y_k)}{\Pr(x_1, \ldots, x_m)}$$

– Assume conditional independence of features $A_1, \ldots, A_m$ given Y. Then

$$\Pr(x_1, x_2, \ldots, x_m | y_k) \overset{\text{product rule}}{=} \prod_{j=1}^{m} \Pr(x_j | x_1, x_2, \ldots, x_{j-1}, y_k) \overset{\text{c. i. a.}}{=}$$

$$= \prod_{j=1}^{m} \Pr(x_j | y_k)$$

– $\Pr(x_1, \ldots, x_m)$ is constant. Then

$$y^\star = \text{argmax}_{y_k \in Y} \Pr(y_k) \prod_{j=1}^{m} \Pr(x_j | y_k) \tag{5}$$

Discriminative vs. generative classifiers

Computing $\Pr(y|\mathbf{x})$

- **discriminative classifier** does not care about how the data was generated. It directly discriminates the value of y for any $\mathbf{x}$.
- **generative classifier** models how the data was generated in order to classify an example.

Discriminative vs. generative classifiers

- Logistic regression classifier is a **discriminative classifier**

$$f(\mathbf{x}; \Theta) = p(y = 1|\mathbf{x}, \Theta)$$

- Naïve Bayes classifier is a **generative classifier**

1. Learn $\Pr(\mathbf{x}|y)$ and $\Pr(y)$
2. Apply Bayes rule to get

$$\Pr(y|\mathbf{x}) = \frac{\Pr(\mathbf{x}|y) \cdot \Pr(y)}{\Pr(\mathbf{x})} \sim \Pr(\mathbf{x}|y) \cdot \Pr(y)$$

3. Classify $\mathbf{x}$

$$y^\star = \operatorname{argmax}_y \Pr(y|\mathbf{x}) = \operatorname{argmax}_y \Pr(\mathbf{x}|y) \cdot \Pr(y)$$

Naïve Bayes classifier

Naive assumption of feature conditional independence given a target class is rarely true in real world applications (high bias). Nevertheless, Naïve Bayes classifier surprisingly often shows good performance in classification (low variance).

Bias-Variance Trade-off -> next lecture

Naïve Bayes Classifier is a linear classifier

NB classifier gives a method for predicting rather than for building an explicit classifier.

Let us focus on **binary classification** $Y = \{0, 1\}$ with binary features $A_1, \ldots, A_m$.

We predict 1 iff

$$\frac{\Pr(y = 1) \prod_{j=1}^{m} \Pr(x_j | y = 1)}{\Pr(y = 0) \prod_{j=1}^{m} \Pr(x_j | y = 0)} > 1$$

Naïve Bayes Classifier is a linear classifier

Denote $p_j = \Pr(x_j = 1|y = 1)$, $q_j = \Pr(x_j = 1|y = 0)$

Then

$$\frac{\Pr(y=1)\prod_{j=1}^{m} p_j^{x_j}(1-p_j)^{1-x_j}}{\Pr(y=0)\prod_{j=1}^{m} q_j^{x_j}(1-q_j)^{1-x_j}} > 1$$

$$\frac{\Pr(y=1)\prod_{j=1}^{m}(1-p_j)(\frac{p_j}{1-p_j})^{x_j}}{\Pr(y=0)\prod_{j=1}^{m}(1-q_j)(\frac{q_j}{1-q_j})^{x_j}} > 1$$

Naïve Bayes Classifier is a linear classifier

Take logarithm

$$\log \frac{\Pr(y=1)}{\Pr(y=0)} + \sum_{j=1}^{m} \log \frac{1-p_j}{1-q_j} + \sum_{j=1}^{m} (\log \frac{p_j}{1-p_j} - \log \frac{q_j}{1-q_j}) x_j > 0$$

NB classifier as a linear classifier where

$$\theta_0 = \log \frac{\Pr(y=1)}{\Pr(y=0)} + \sum_{j=1}^{m} \log \frac{1-p_j}{1-q_j}$$

$$\theta_j = \log \frac{p_j}{1-p_j} - \log \frac{q_j}{1-q_j}, \quad j = 1, \ldots, m$$

Bayesian belief networks (BBN)

- Naïve Bayes classifier assumes that ALL features are conditionally independent given a target attribute.
- A Bayesian network is a probabilistic graphical model that encodes probabilistic relationships among attributes of interest.
- BBNs allow stating conditional independence assumptions that apply to subsets of the attributes.
- Dependencies are modeled as graph where nodes correspond to attributes and edges to dependency between attributes.

Bayesian belief networks
Settings

Consider an arbitrary set of random variables $X_1, X_2, ..., X_m$. Each variable X_i can take on the set of possible values $Values(X_i)$.

We define the **joint space** of the variables $X_1, X_2, ..., X_m$ to be the cross product $Values(X_1) \times Values(X_2) \times Values(X_3) \times ... \times Values(X_m)$.

The probability distribution over the joint space is called the **joint probability distribution** $\Pr(x_1, x_2, ..., x_m)$ where
$x_1 \in Values(X_1), x_2 \in Values(X_2), ..., x_n \in Values(X_m)$.

BBN describes the joint probability distribution for a set of variables by specifying a set of conditional independence assumptions together with sets of local conditional probabilities.

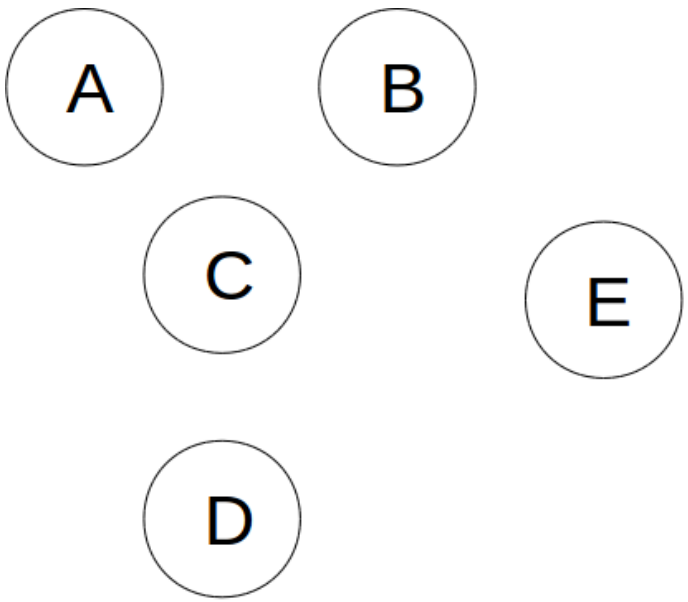
Bayesian belief networks

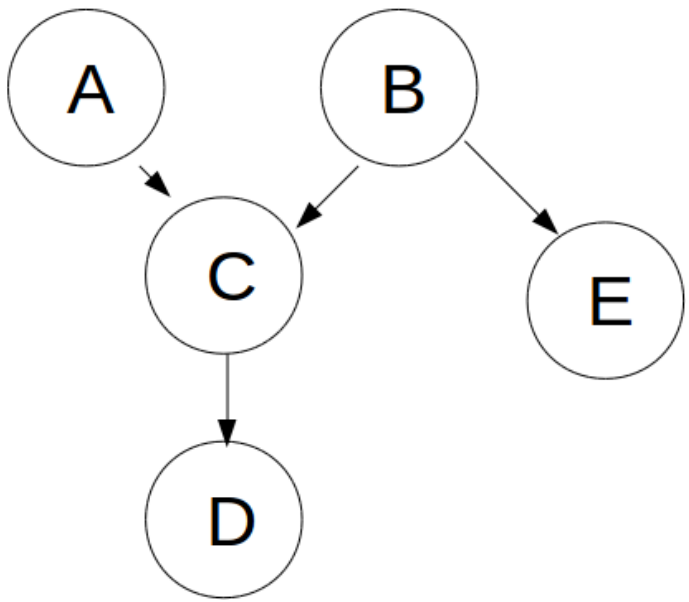
Representation

1. A directed acyclic graph $G = (V, E)$
   - nodes are random variables
   - arcs between nodes represent probabilistic dependencies
   - Y is a *descendant* of X if there is a directed path from X to Y
2. The network arcs represent the assertion that the variable X is conditionally independent of its nondescendants given its immediate predecessors $Parents(X)$; $\Pr(X|Parents(X))$
3. A set of tables for each node in the graph - a conditional probability table is given for each variable; it describes the probability distribution for that variable given the values of its immediate predecessors.

Building a Bayes net

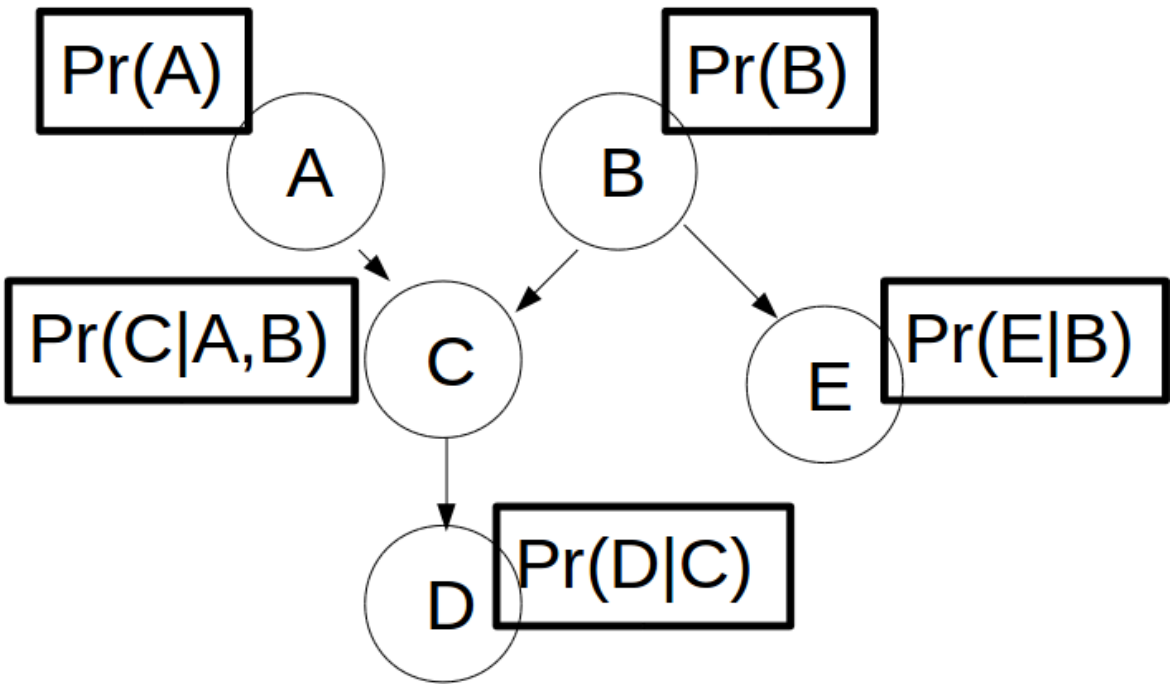
1. Choose the variables to be included in the net: A, B, C, D, E
2. Add the links





Building a Bayes net

3. Add a probability table for each root node $\Pr(X)$ and nonroot node $\Pr(X|Parents(X))$



Once the net is built ...

The join probability of any assignment of values $x_1, x_2, ..., x_m$ to the tuple of network variables $X_1, X_2, ..., X_m$ can be computed by the formula

$$\Pr(x_1, x_2, ..., x_m) = \Pr(X_1 = x_1 \wedge X_2 = x_2 \wedge \cdots \wedge X_m = x_m) = \prod_{i=1}^{m} \Pr(x_i | Parents(X_i)) \quad (6)$$

Bayesian belief networks

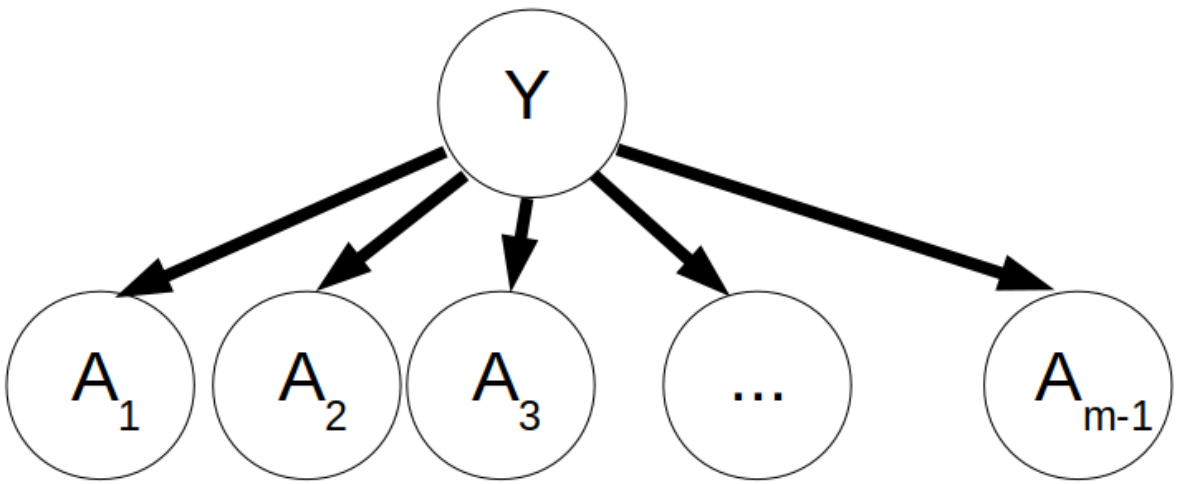
Two components

1. A function for evaluating a given network based on the data.
2. A method for searching through the space of possible networks.

Learning the network structure

- searching through the space of possible sets of edges
- estimating the conditional probability tables for each set
- computing the quality of the network

Bayesian belief networks
Naïve Bayes Classifier



K2 algorithm

This 'search and score' algorithm heuristically searches for the most probable belief-network structure given a training data.

It starts by assuming that a node has no parents, after which, in every step it adds incrementally the parent whose addition mostly increase the probability of the resulting structure. K2 stops adding parents to the nodes when the addition of a single parent cannot increase the probability of the network given the data.

Summary of Examination Requirements

- Hyperplane, margin, functional margin, geometric margin of example and data set
- Large margin classifier
  linearly separable data, supporting hyperplanes, support vectors, optimization task, prediction function
- Soft margin classifier
  not linearly separable data, supporting hyperplanes, support vectors, slack variables, optimization task, hyperparameter C, prediction function
- Kernel trick
  feature mapping, Kernel functions, prediction function
- Discriminative and generative classifiers
- Naïve Bayes Classifier
  conditional independence, linear decision boundary
- Bayesian networks
  structure, conditional probabilities