Outline

• Naïve Bayes algorithm

• Bayesian networks
Bayes theorem

**Probabilistic approach to classification** $Y = \{y_1, y_2, \ldots, y_K\}$

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

**Bayes theorem**

$$\text{posterior probability} = \frac{\text{prior probability} \times \text{likelihood}}{\text{marginal likelihood}} \hspace{1cm} (2)$$

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

Then

$$y^* = \text{argmax}_{y_k \in Y} \frac{\Pr(y_k) \cdot \Pr(x_1, \ldots, x_m|y_k)}{\Pr(x_1, \ldots, x_m)}$$  \hspace{1cm} (3)
Let $X$, $Y$ and $Z$ be three discrete random variables. We say that $X$ is \textit{conditionally independent} of $Y$ given $Z$ if

\[ \forall x_i, y_j, z_k, x_i \in \text{Values}(X), y_j \in \text{Values}(Y), z_k \in \text{Values}(Z) : \]

\[ \Pr(X = x_i | Y = y_j, Z = z_k) = \Pr(X = x_i | Z = z_k) \]  \hspace{1cm} (4)

I.e., $P(X|Y, Z) = P(X|Z)$. 
Conditional independence

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

<table>
<thead>
<tr>
<th>Sky</th>
<th>AirTemp</th>
<th>Humidity</th>
<th>Wind</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>warm</td>
<td>normal</td>
<td>strong</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>Yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cold</td>
<td>high</td>
<td>strong</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>warm</td>
<td>high</td>
<td>strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

\[ y^* = \arg\max_{y_k \in \mathcal{Y}} \Pr(y_k | x_1, \ldots, x_m) = \arg\max_{y_k \in \mathcal{Y}} \frac{\Pr(y_k) \cdot \Pr(x_1, \ldots, x_m | y_k)}{\Pr(x_1, \ldots, x_m)} \]

– 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

\[ y^* = \arg\max_{y_k \in \mathcal{Y}} \Pr(y_k) \prod_{j=1}^{m} \Pr(x_j | y_k) \] (5)
Computing $\Pr(y|x)$

- **discriminative classifier** does not care about how the data was generated. It directly discriminates the value of $y$ for any $x$.

- **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**
  \[ f(x; \Theta) = p(y = 1|x, \Theta) \]

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

  ₁ Learn \( \Pr(x|y) \) and \( \Pr(y) \)

  ₂ Apply Bayes rule to get
  \[
  \Pr(y|x) = \frac{\Pr(x|y) \cdot \Pr(y)}{\Pr(x)} \sim \Pr(x|y) \cdot \Pr(y)
  \]

  ₃ Classify \( x \)
  \[ y^* = \arg\max_y \Pr(y|x) = \arg\max_y \Pr(x|y) \cdot \Pr(y) \]
Naïve 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

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

\[
\frac{\Pr(y = 1) \prod_{j=1}^{m} (1 - p_j)\left(\frac{p_j}{1-p_j}\right)^{x_j}}{\Pr(y = 0) \prod_{j=1}^{m} (1 - q_j)\left(\frac{q_j}{1-q_j}\right)^{x_j}} > 1
\]
Naïve Bayes Classifier is a linear classifier

Take logarithm

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

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.
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 $\text{Values}(X_i)$.

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

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

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 $\text{Parents}(X)$; $\Pr(X|\text{Parents}(X))$

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.
1. Choose the variables to be included in the net: $A, B, C, D, E$
2. Add the links
3. Add a probability table for each root node $Pr(X)$ and nonroot node $Pr(X|\text{Parents}(X))$
Once the net is built ...

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

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

(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

![Diagram of a Bayesian belief network with nodes Y, A₁, A₂, A₃, ..., Aₘ₋₁ connected with arrows.]
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