Bayesian Networks have been applied in many different scenarios!
Learning and Inference in Bayesian Networks
Learning and Inference

Two main tasks when dealing with Bayesian Networks

- **Learning**: the problem of “building” the model
  - how to set the parameters? Typical solution: to exploit a set of objects sampled from the problem to estimate them (learning from examples)

- **Inference**: the problem of “querying” the model
  - Compute interesting probabilistic relations or values of variables in the model. Typically the inference is performed once the model is learnt
Learning

Example 1: the “Two Box problem”

- There are two boxes, one red and one blue
- The boxes are covered with a blanket, so that we cannot observe the color
- In the red box there are 2 apples and 6 oranges, in the blue box there are 3 apples and 1 orange
- We want to model the procedure of extracting fruits from the boxes (with re-integration of the fruit in the box)
The problem is to estimate the parameters $\alpha$, $\beta$, and $\gamma$. 

Learning
Learning

\[ \alpha \]

\[ P(B) \]

\[ P(B = 'b') = \alpha \]
\[ P(B = 'r') = 1 - \alpha \]

\( \alpha \) represents the probability of choosing the blu box

We can assume that the two boxes are equivalent (i.e. \( \alpha = 0.5 \))
Learning

- \( \alpha \)
- \( \beta \) represents the probability of extracting the orange from the red box
- In the red box there are 6 oranges over 8 fruits
- \( \beta = 0.75 \)

\[
P(F|B) =
\begin{align*}
P(F = 'o' | 'B = 'r') &= \beta \\
P(F = 'a' | 'B = 'r') &= 1-\beta \\
P(F = 'o' | 'B = 'b') &= \gamma \\
P(F = 'a' | 'B = 'b') &= 1-\gamma
\end{align*}
\]
Learning

\( \gamma \) represents the probability of extracting the orange from the blue box.

In the blue box there is 1 orange over 4 fruits.

\( \gamma = 0.25 \)
Learning

- In this case we learn the model using a priori knowledge (information we have)
  - This is not the case in many contexts!
- In alternative, the learning can be carried out **automatically** by using the “learning from example” paradigm
  - We sample some objects from the problem (**Training set**), usable to estimate the parameters
- This represents a difficult task, especially if the BN contains hidden variables
Learning

**Training set:** some objects sampled from the problem.

In this case we don't have any a priori information on the content of the two boxes, we only observe the fruits (oranges/apples) – the *visible* variables!

Problem: we don't know from which box these fruits derive (the variable B is *hidden*)
The estimation of $\gamma$ and $\beta$ can be complicated!
Learning

- To learn Bayesian Networks with hidden variables we can resort to the Expectation – Maximization algorithm
- This represents a widely applied method to perform a Maximum Likelihood estimation of the parameters of a probabilistic model

.. let's briefly summarize the Maximum Likelihood estimation
Maximum Likelihood estimation

- The problem: given the training set $D=\{x_1..x_N\}$ (which contains objects sampled from the problem), the goal is to learn the model (i.e. the Bayesian Network), i.e. to estimate the parameters (of the BN)

- Let's call this set of parameters $\theta$
  - In the “two boxes” problem, $\theta=\{\alpha,\beta,\gamma\}$

- Please note that, if we know $\theta$ the model is completely specified (we can compute $p(x)$ for all objects)

- Goal: estimation of $\theta$ from $D$
Maximum Likelihood estimation

- Starting point: the likelihood function $P(D|\theta)$
  - It represents the joint probability of all points in the training set
  - It clearly depends on the choice of the parameter $\theta$ (this is why it is denoted as $P(D|\theta)$)
- Assuming that all points in the training set are i.i.d (independent and identically distributed), the likelihood is defined as

$$P(D|\theta) = \prod_{i=1}^{N} p(x_i|\theta)$$
Example: 1D Gaussian, \( D \) contains 3 points

\[ D = \{x_1, x_2, x_3\} \]

\[ p(x \mid \theta) = N(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{||x-\mu||^2}{2\sigma^2}} \]

\[ \theta = \{\mu, \sigma\} \]

\[ P(D \mid \theta) = p(x_1, x_2, x_3 \mid \theta) = p(x_1 \mid \theta)p(x_2 \mid \theta)p(x_3 \mid \theta) \]

\( x_1, x_2 \) and \( x_3 \) are independent and identically distributed (i.e. they follow the same distribution, the Gaussian)
**Note:** if we change the parameters $\theta$, also the likelihood changes!

\[ P(D|\theta) = p(x_1, x_2, x_3|\theta) = p(x_1|\theta)p(x_2|\theta)p(x_3|\theta) \]

In this case the likelihood is larger: the dataset is explained in a better way by the Gaussian identified by this $\theta$!
Maximum Likelihood estimation

- Given a parameter $\theta$, we can consider the likelihood $P(D|\theta)$ as a measure of “how well the training set is explained by the model defined by the parameter $\theta$”
- The likelihood $P(D|\theta)$ is a function of $\theta$ ($D$ is fixed)

The Maximum Likelihood estimate of the parameter of the model is defined as the parameter $\theta_{ML}$ which maximizes the likelihood $P(D|\theta)$

$$\theta_{ML} = \arg \max_{\theta} P(D|\theta)$$
Maximum Likelihood estimation

- **Example**: \( \mathbf{D} \) contains some points to be modelled with a one-dimensional Gaussian
  - The variance is known, the only parameter is the mean (\( \theta = \{\mu\} \))
Maximum Likelihood estimation

Every Gaussian is identified by a specific value of the parameter $\theta$ (i.e. the mean)

4 out of all the possible Gaussians (infinite)
Maximum Likelihood estimation

Let's compute the likelihood, i.e. “how well the training set is explained by the model defined by a specific parameter $\theta$”

This parameter (which identifies a Gaussian) does not explain well the dataset $\rightarrow$ low likelihood
Maximum Likelihood estimation

This parameter represents a better choice → the likelihood is higher!
Maximum Likelihood estimation

This parameter represents the best choice: the corresponding Gaussian is the one explaining in the best way the dataset → maximum of the likelihood!
Maximum Likelihood estimation

- Important note: the likelihood $P(D|\theta)$ is a **function of $\theta$**, whereas the probability $p(x|\theta)$ is a **function of $x$** ($p(x|\theta)$ is the probability of a point given a model specified with the parameter $\theta$)
- In order to maximize the likelihood, it is often mathematically convenient to work with the logarithm of the likelihood, often called **log-likelihood**

$$l(\theta) = \ln P(D|\theta) = \ln \prod_{i=1}^{N} p(x_i|\theta) = \sum_{i=1}^{N} \ln P(x_i|\theta)$$
Maximum Likelihood estimation

The $\theta$ maximizing $P(D|\theta)$ also maximizes $l(\theta)$

The logarithm is a monotonic function

This trick is useful when we have to compute derivatives to maximize the likelihood (the derivative of a sum is more manageable than the derivative of a product)
Maximum Likelihood estimation

- In many cases the ML estimate provides a **sufficiently good** value for the parameter
  - This is especially true when the training set is reasonably **large**
- Alternatives exist: the **Bayesian estimation** (not seen here)
  - Instead of considering that there exists a **single** optimal parameter \( \theta \) which permits to explain the dataset, the Bayes estimate takes into account **all the possible values** of \( \theta \) to define the \( p(x) \), each one with its own probability
Bayesian estimation

\[ p(x|D) = \int p(x|\theta)p(\theta|D)d\theta = \int p(x|\theta)p(D|\theta)p(\theta)d\theta \]

- Powerful approach, but difficult in practical applications
  - computing the integral may be impossible (intractable integral)
  - defining the priors \( p(\theta) \) may be problematic
The Expectation – Maximization (EM) algorithm

- The Maximum Likelihood estimation requires to optimize the Likelihood (or the log likelihood)
- Depending on the form of $p(x|\theta)$ the maximization can be easy or difficult
  - For example: if $p(x|\theta)$ is a Gaussian, we can set the derivatives of $l(\theta)$ to zero, and solve for $\theta$
- However, for many models the analytical solution can not be retrieved, and we have to resort to more complex technique, such as the Expectation-Maximization (E-M) algorithm
The EM algorithm

- This represents a general method of finding, from a given data set, the maximum-likelihood estimate of the parameters of a probabilistic model with hidden variables.
- EM is an iterative algorithm which, at each iteration, is guaranteed to increase the likelihood.
- It is also guaranteed to converge to a maximum of the likelihood.
  - The obtained maximum is however local.
The EM algorithm

- General formulation

(Board)
The EM algorithm

 Initialization: set $\theta^{(0)}$

 Repeat:
  E-Step: compute the $Q(\theta, \theta^{(i-1)})$ function
  \[
  Q(\theta, \theta^{(i-1)}) = E \left[ \log p(X, Y | \theta) | X, \theta^{(i-1)} \right]
  \]

  M-Step: re-estimate the parameters
  \[
  \theta^i = \arg \max_{\theta} Q(\theta, \theta^{(i-1)})
  \]

 Until convergence (e.g. $\theta^i - \theta^{(i-1)} < \text{epsilon}$)
The EM algorithm

- EM for mixtures of Gaussians

(Board)
The EM algorithm

- In summary: in the E-STEP the EM estimates the probability that every Gaussian had generated the different points

For every point $x_i$

\[ w_{i1} = p(y_i = 1 | \mathcal{X}, \theta^{(i-1)}) = p(y_i = 'blue' | \mathcal{X}, \theta^{(i-1)}) \]

$w_{i1}$ represents how probable is that the point has been generated by the first Gaussian given the current parameters

\[ w_{i2} = p(y_i = 2 | \mathcal{X}, \theta^{(i-1)}) = p(y_i = 'red' | \mathcal{X}, \theta^{(i-1)}) \]
Current guesses

blue point: $w_{i1} > w_{i2}$! It is more probable that the point has been generated by the blue Gaussian

Red point: $w_{i2} > w_{i1}$! It is more probable that the point has been generated by the red Gaussian

This point is purple (a mixture of blue and red): $w_{i1} \sim w_{i2}$! Both Gaussians are equiprobable

Note: $w_{i1} + w_{i2} = 1$
The EM algorithm

- in the M-STEP the EM re-estimates the parameters using the values computed in the E-STEP

\[ \mu_1 = \sum_{i=1}^{N} w'_{i1} x_i \]

The new mean for the Blue Gaussian is the average of all points, each one weighted with the probability of having been generated by the blue Gaussian.

\[
\begin{align*}
    w'_{i1} &= \frac{w_{i1}}{\sum_{j=1}^{N} w_{j1}}
\end{align*}
\]

Note: In the classical average: all points have the same weight (\(w_i = 1/N\))

\[
\text{Average} = \frac{1}{N} \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} \frac{1}{N} x_i
\]

Weighted average: each point contributes according to the weight

\[
\text{Weighted Average} = \sum_{i=1}^{N} w_i x_i
\]
For these points, $w_{i1}$ is larger than $w_{i2}$, therefore they would contribute more to the novel estimation of the blue Gaussian.

For these points, on the contrary, $w_{i2}$ is larger, therefore they would contribute more to the novel estimation of the red Gaussian.
The EM algorithm, starting from an initial estimate, converges to a local optimum.

Since typically the Log likelihood is highly multimodal, finding a good starting point is crucial to get a good estimate.
Summary

- The EM algorithm represents a flexible tool to learn a Maximum Likelihood estimate of the parameters of a Bayesian Network
  - It can be used also in other contexts
- It requires complex mathematical derivations: depending on the complexity of the Bayes Net, it can be easy – difficult – impossible to derive analytical E-Step and M-Step
- Needed tradeoff: **computability** vs **descriptivity**