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

\[ \gamma \rightarrow B \rightarrow F \rightarrow \beta \]

- \( \alpha \) represents the probability of choosing the blu box.
- We can assume that the two boxes are equivalent (i.e., \( \alpha = 0.5 \)).

\[
\begin{align*}
P(B) &= \alpha \\
P(B = 'b') &= \alpha \\
P(B = 'r') &= 1 - \alpha
\end{align*}
\]
Learning

\( \beta \) represents the probability of extracting the orange from the red box.

In the red box there are 6 oranges over 8 fruits.

\( \beta = 0.75 \)
Learning

[diagram showing a learning model with nodes labeled α, B, γ, F, and β, connected by arrows]

\[ P(F|B) \]

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

\( \gamma \) 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
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 $\mathbf{D}=\{x_1..x_N\}$ (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 $\mathbf{D}$
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

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

\( \theta = \{\mu, \sigma\} \)

\( D = \{x_1, x_2, x_3\} \)

\( x_1, x_2, \) 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!

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:** $D$ 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 \( \theta \)”

![Diagram showing Maximum Likelihood estimation](image)

This parameter (which identifies a Gaussian) does not explain well the dataset → 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 $P(\mathcal{D}|\theta)$ 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

- Powerful approach, but difficult in practical applications
  - computing the integral may be impossible (intractable integral)
  - defining the priors $p(\theta)$ may be problematic

\[
p(x|D) = \int_{\theta} p(x|\theta)p(\theta|D)d\theta = \int_{\theta} p(x|\theta)p(D|\theta)p(\theta)d\theta
\]
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
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. $\theta^{(i)} - \theta^{(i-1)} < \epsilon$)
The EM algorithm

- EM for mixtures of Gaussians
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 $x_i$

$$w_{i1} = p(y_i = 1|\mathcal{X}, \theta^{(i-1)}) = p(y_i = \text{\textquoteleft blue\textquoteright}|\mathcal{X}, \theta^{(i-1)})$$

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

$$w_{i2} = p(y_i = 2|\mathcal{X}, \theta^{(i-1)}) = p(y_i = \text{\textquoteleft red\textquoteright}|\mathcal{X}, \theta^{(i-1)})$$
The EM algorithm

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

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

\[ w'_{i1} = \frac{w_{i1}}{\sum_{j=1}^{N} w_{j1}} \]

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.

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

\[ \text{Weighted Average} = \sum_{i=1}^{N} w_i x_i \]
Current guesses $\theta^{(i-1)}$

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

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

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

Note: $w_{i1} + w_{i2} = 1$
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.
Initialization

- 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**
Inference

• Once the model is learnt, it can be interesting to **query** the model, i.e. to ask something
• Example 1: the “**Two Box problem**”
Inference

- A possible interesting question can be: “What is the probability of extracting an apple?”
  - I.e. we are interested in $P(F = 'a')$
- This probability is obtained by exploiting the following facts
  - $P(F)$ can be obtained from $P(B,F)$ by marginalization
  - $P(B,F)$ – the joint probability – is given by the Bayesian Network, which also provides a factorization of it
Inference

The Bayesian Network provides a factorization of the joint probability

\[ P(F) = \sum_{B} P(F, B) \]
\[ = \sum_{B} P(B)P(F|B) \]
Inference

\[
P(F \neq 'a') = \sum_{B} P(F \neq 'a', B) \\
= \sum_{B} P(B)P(F \neq 'a'|B) \\
= P(B = 'b')P(F \neq 'a'|B = 'b') \\
+ P(B = 'r')P(F \neq 'a'|B = 'r') \\
= \alpha(1 - \gamma) + (1 - \alpha)(1 - \beta) \\
= 0.5 \cdot 0.75 + 0.5 \cdot 0.25 \\
= 0.5
\]
Inference

- **Exercise**: compute $p(x = 5)$ within a Gaussian Mixture Model with 2 one-dimensional Gaussians

\[
p(g = 1) = \pi_1 \quad p(g = 2) = \pi_2
\]

\[
p(x|g = 1) = \mathcal{N}(x|\mu_1, \sigma_1)
p(x|g = 2) = \mathcal{N}(x|\mu_2, \sigma_2)
\]

**Trained GMM**

- $\pi_1 = 0.3, \pi_2 = 0.7$
- $\mu_1 = 4.2, \sigma_1 = 2$
- $\mu_2 = 5.2, \sigma_2 = 1$
**Inference**

\[
P(x = 5) = \sum_{g} P(x = 5, g)
\]

\[
= \sum_{g} P(g)P(x = 5|g)
\]

\[
= P(g = 1)P(x = 5|g = 1) + P(g = 2)P(x = 5|g = 2)
\]

\[
= \pi_1\mathcal{N}(x = 5|\mu_1, \sigma_1) + \pi_2\mathcal{N}(x = 5|\mu_2, \sigma_2)
\]

Knowing that

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

\[
P(x = 5) = 0.3\frac{1}{\sqrt{2\pi\cdot2^2}} e^{-\frac{||5-4.2||^2}{2\cdot2^2}} + 0.7\frac{1}{\sqrt{2\pi\cdot1^2}} e^{-\frac{||5-5.2||^2}{2\cdot1^2}}
\]

\[
= 0.0721 + 0.2737
\]

\[
= 0.3458
\]
Inference

- Another interesting inference can be performed on the hidden variables
  - To understand the causes
- Two boxes example: knowing that I have extracted an apple, what is the probability that I've chosen the blue box?
  - $P(B = 'b' \mid F = 'a')$
- NOTE: $P(B = 'b' \mid F = 'a')$ is different from the conditional of the Bayesian network – $P( F = 'a' \mid B = 'b')$
Inference

- The procedure is similar

\[ P(B|F) = \frac{P(F, B)}{P(F)} \]

Definition of conditional probability

\[ P(F, B) = P(B)P(F|B) \]

The joint probability (and its factorization) is given by BN

\[ P(F) = \sum_B P(F, B) = \sum_B P(B)P(F|B) \]

P(F) is computed as before via marginalization
Other inferences

- Inference needed during **learning**
  - Example: the E-M for mixture of Gaussians

  posterior to be computed in the E-Step for the hidden variable

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

- **Optimization**: which is the configuration of the hidden variables for which the joint probability is **maximum**? Which is the **most probable** configuration of the hidden variables?
  - Often called MAP (Maximum a Posteriori) estimation
Inference: summary

- Inference is used to extract interesting information from the Bayesian Network
  - Summaries, causes, optimization
- In general, performing inference is not always so easy (e.g. integrals)

Example: a GMM with an infinite number of components

\[ p(x) = \int_g p(x, g) dg \]
Inference: summary

- Another aspect: the structural complexity of Bayesian Networks (e.g. cycles) may make the inference problem intractable

- (Again) Needed tradeoff: **computability** vs **descriptivity**
Inference: summary

Many complex algorithms have been proposed to perform non-trivial inference (not seen here):

- Exact inference (variable elimination, belief propagation – for trees, ..)
- Variational inference (mean field)
- Monte Carlo inference (Gibbs sampling)

For more info see the Kevin Murphy's tutorial (further readings)
Other Probabilistic Graphical Models

- There are other two families of Probabilistic Graphical Models:
  - Markov Random Fields
  - Factor Graphs
Markov Random Fields

- **Undirected** graph of random variables
- Each variable is independent of all other variables, given its neighbors
Markov Random Fields

- Widely used to analyse images

They can model continuity (smoothness) and spatial proximity, crucial aspects in images

Example: noise removal
Factor Graphs

- Less investigated class of probabilistic Graphical Models (introduced by Frey)
- Main idea: to express a global function of several variables as a collection of factors (local functions) over a subset of those variables
- The graph has two kinds of node:
  - Nodes for **variables**
  - Nodes for functions (called **factors**)

Factor Graphs

$x_1, x_2, x_3$ are the variables
$f_a, f_b, f_c, \text{ and } f_d$ are factors

This factor graph encodes the factorization of a function $g(x_1, x_2, x_3)$ over all the variables

$$g(x_1, x_2, x_3) = f_a(x_1, x_2) f_b(x_3) f_c(x_2, x_3) f_d(x_2, x_3)$$
Factor Graphs

- This formalism is more general than the Bayesian Network formalism

Every Bayesian Network can be written as a Factor Graph

\[ P(B, F) = P(B)P(F|B) \]

\[ f_1(B) = P(B) \]
\[ f_2(B, F) = P(F|B) \]
\[ g(B, F) = f_1(B)f_2(B, F) = P(B)P(F|B) = P(B, F) \]
Factor Graphs

- This formalism can be used also for “non probabilistic” functions

Example: **Affinity Propagation** algorithm for Clustering

*Science 315, 972–976 (2007)*
Conclusions

- Probabilistic Graphical Models represent a powerful tool to model structured objects
  - Capability to capture the complexity
  - Different information can be extracted
  - Many algorithms / tools to perform training, inference, optimization

- Constraint: tradeoff between *computability* and *descriptivity*
Further Readings