Probabilistic Graphical Models | 3

3.1 Probabilistic Inference

3.1.1 Introduction

Suppose to have the following set of random variates, which can be either discrete or continuous:

- $x = (x_1, \dots, x_n)$
- $z = (z_1, \dots, z_m)$

A **probabilistic model of the world** is expressed as a joint probability distribution, i.e. $p(x, z)$.

Typically, to reason about such a model, we would like to perform these operations:

- Marginalization: $p(x) = \int p(x,z)dz$ (or $\sum_z p(x,z)$ if z is a discrete R.V.)
- Conditioning: $p(z|x = \bar{x}) = \frac{p(\bar{x},z)}{p(\bar{x})}$, where $\bar{x}$ is a fixed value

These two operations may be combined, i.e. we may want to compute

$$p(z_j|x_i = \bar{x}_i) = \frac{p(\bar{x}_i, z_j)}{p(\bar{x})} = \frac{\int p(x,z)dx_{-i}dz_{-j}}{\int p(x,z)dx_{-i}dz}$$

What is the cost in terms of computation of performing these operations?

Let's focus on discrete values for z such that $z_i \in \{0, 1\}$ and consider the marginalization over z:

$$p(x) = \sum_{z \in Z} p(x, z)$$

This seems like an easy task, but $z \in Z$, with $|Z| = 2^m$. If $m = 100$, then $|Z| \approx 10^{30}$, and we would have no chance to compute the summation, or even store our joint distribution in memory. Therefore knowing our model of the world might not be enough to get information regarding the world.

3.1.2 Factorization

Consider $p(x_1, \dots, x_n)$. Applying the laws of probability we can write

$$\begin{aligned}
p(x_1 \dots, x_n) = \quad & \text{(a)} \\
p(x_2, \dots, x_n|x_1)p(x_1) = \quad & \\
p(x_3, \dots, x_n|x_1, x_2)p(x_2|x_1)p(x_1) = \quad & \text{(b)} \\
\dots \quad & \\
p(x_n|x_1, \dots x_{n-1})p(x_{n-1}|x_1, \dots, x_{n-2}) \dots p(x_3|x_2, x_1)p(x_2|x_1)p(x_1) &
\end{aligned}$$

This is known as a **factorization** of our joint distribution. In some cases, factorization can be simpler, because some variables in the conditioning set of a factor may be irrelevant to other variables. For example we could have

$$p(x_1, \cdots, x_n) = p(x_n|x_{n-1})p(x_{n-1}|x_{n-2}) \cdots p(x_2|x_1)p(x_1) \qquad \text{(c)}$$

What is the cost in terms of computer memory, to store these three representation of our distribution? Suppose $x_i \in \{1, \cdots, k\}$. Then we have the following storing costs:

- $p(x_1)$ costs $k - 1 = O(k)$ floating point numbers (either in single or double precision)
- $p(x_2|x_1)$ costs $k(k-1) = O(k^2)$ float numbers (we have k different distributions for x_2, each costing $k - 1$ floats)
- In general $p(x_n|x_1, \ldots x_{n-1})$ costs $O(k^n)$ float numbers
- Factorization (a) costs k^n float numbers. Unfeasible.
- Factorization (b) costs $O(k^n)$ because of the term $p(x_n|x_{n-1}, \cdots x_1)$
- Factorization (c) instead has just a total cost of $O(nk^2)$. This makes storing it in memory tractable.

Therefore, if we are able to exploit factorizations in some way, the problems of conditioning and marginalization become tractable.

Remark: given a joint distribution $p(x_1, \ldots, x_n)$, there are several ways (in fact, $n!$) of factorizing it, similarly to (b) above, depending on the order of variables. Any of these choices may correspond to a different factorization, i.e. to different ways of removing variables from conditioning set. The problem of identifying the most parsimonious factorization is known to be NP-hard.

Probabilistic Graphical Models (PGM) are models of a probability distribution that describe/ impose a certain factorization, hence allowing us to store and make inference effectively with joint distributions.
There are three main kind of PGM:

- Bayesian networks
- Markov Random Fields
- Factor Graphs

3.2 Bayesian Networks

Consider a joint distribution on $x_1, \ldots, x_4$, and assume the following factorization holds:

$$p(x_1, x_2, x_3, x_4) = p(x_4|x_3)p(x_3|x_2, x_1)p(x_2)p(x_1)$$

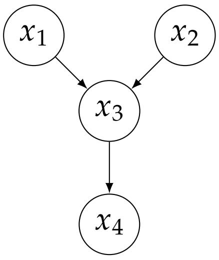
Definition 3.2.1 *We call pa_k the parents of x_k, i.e., the conditioning set of x_k*

This means that $p(x_k|pa_k)$ is the factor for x_k.

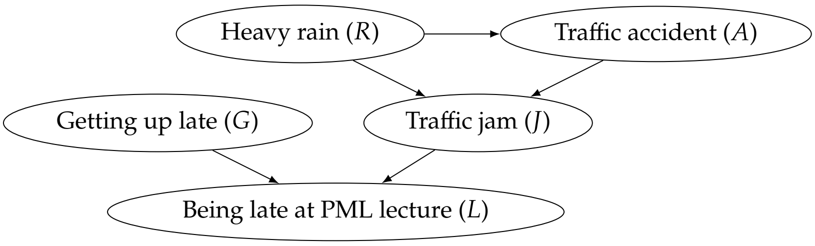
Example: In $p(x_1, x_2, x_3, x_4) = p(x_4|x_3)p(x_3|x_2, x_1)p(x_2)p(x_1)$ we have: $pa_4 = \{x_3\}, pa_3 = \{x_2, x_1\}, pa_2 = \{\}, pa_1 = \{\}$

Definition 3.2.2 *A* ***Bayesian Network (BN)*** *is a DAG (Direct Acyclic graph), having* $\{x_1, \dots, x_n\}$ *as vertices and* (x_i, x_j)*,* $i < j$ *and* $x_i \in pa_j$ *as edges.*

Example: The BN for $p(x_1, x_2, x_3, x_4) = p(x_4|x_3)p(x_3|x_2, x_1)p(x_2)p(x_1)$ is:



Example:

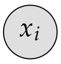


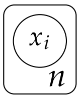
This BN corresponds to the following factorization:

$$p(L, G, J, R, A) = p(L|G, J)p(J|R, A)p(A|R)p(R)p(G)$$

3.2.1 Notational conventions

There are some conventions for writing the BN graph, here we list the most common ones:

- Observed nodes (Random variables of which we know the value) are **coloured** or shadowed 

- Plated nodes are a shorthand for a collection of nodes. The following represents the nodes $x_1, \dots, x_n$ 

- Deterministic quantities represented as small solid circles. These are fixed parameters of the model (represented graphically as $\bullet\ \alpha$)

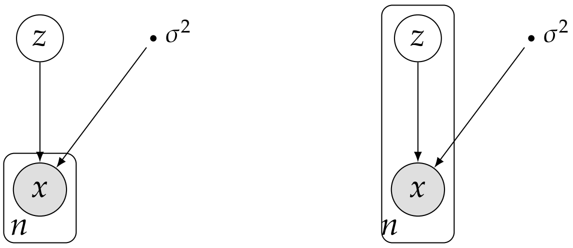
Example: Suppose to have now a RV z that follows a discrete distribution and another RV x that is normally distributed with mean $\mu(z)$, depending on the value of z, and variance σ^2 (shorthand for this is $x \sim \mathcal{N}(\mu_z, \sigma^2)$. This model is known as a **mixture of gaussians**. We can write

$$p(x, z) = p(x|z)p(z)$$

with $p(x|z) = \mathcal{N}(x|\mu_z, \sigma^2)$.

Typically, in this scenario we have at our disposal n observations of the variable x, $\bar{x} = x_1, \cdots, x_n$, while the corresponding variable z is unobserved (hence z is a **latent** variable).We typically want to compute $p(z|\bar{x})$.

We can have two scenarios here: in the first one, $\bar{x}$ are sampled from a single realization of the variable z (on the left), while in the second one each x_i may have been drawn with a different z_i, hence we are interested in computing $p(z_i|x_i)$ (on the right):



The parameter σ^2 corresponds to the variance of the normal distribution associated with x.

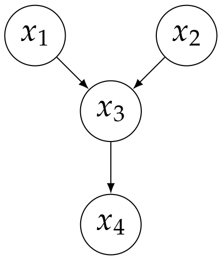
3.3 Sampling and reasoning in Bayesian Networks

We describe now a simple sampling algorithm for BN, and then turn to discuss several reasoning schemes.

3.3.1 Ancestral sampling

Let's recall our example

$$p(x_1, x_2, x_3, x_4) = p(x_4|x_3)p(x_3|x_2, x_1)p(x_2)p(x_1)$$

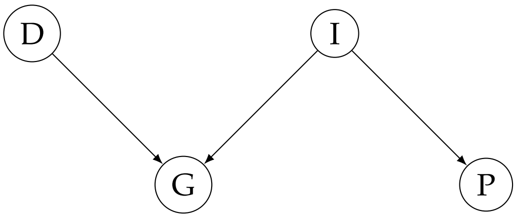


Ancestral sampling is a sampling technique that allows us to sample from a given distribution, if we know its factorization.

One can start to sample from the top of the network (our **ancestrals**), in the example x_1 and x_2, then move to their children and sample, in the example, x_3 from $p(x_3|x_2, x_1)$ using the values for x_1 and x_2 that have *already* been sampled, and so on. It is a simple and effective way to sample probability distribution, provided it is easy to sample from each marginal and conditional.

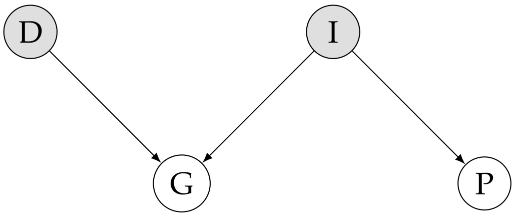
3.3.2 Reasoning with Bayesian Networks: an example

We'll move to another example of a Bayesian Network. We consider a model of students' grade in an exam.



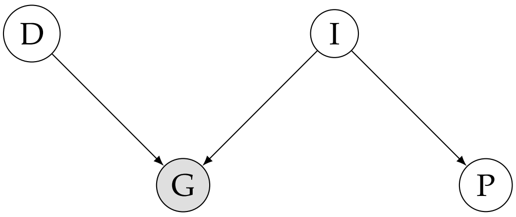
The difficulty of the exam (D) and the intelligence of the student (I) are independent, but the grade you get when you take an exam (G) is dependent on both (studying of course helps too). Having passed a hard exam (P) likely depends on the student's abilities but not on the difficulty and the grade of the current exam. We have three different forms of reasoning here:

- **Causal Reasoning**



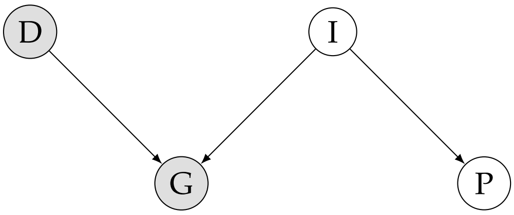
We observe something (the difficulty of the exams, how intelligent we are) and we *infer* the grade and if we have passed a hard exam. The reasoning goes from top to bottom.

- **Evidential reasoning**



If we instead observe the grade, we might want to get information regarding the difficulty of the exams and how intelligent the student is. The reasoning goes from bottom to top.

- **Intercausal reasoning**



The information is propagated in both directions along the network.

Bayesian networks of course can get very large and probabilistic inference becomes a complicated and important task to perform in the real world.

Remark: note that dependency in a BN can model both *causality* and *correlation*, and the two are not necessarily distinguishable within the model.

3.4 Conditional Independence in Bayesian Networks

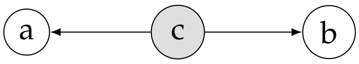
We define now in detail the notion of conditional independence, and study what BNs tell us in this respect.

Definition 3.4.1 *Consider the RVs a, b, c. We say that a is **conditional independent** of b given c (written $a \perp\!\!\!\perp b|c$) iff $p(a|b,c) = p(a|c)$ or equivalently $p(a,b|c) = p(a|c)p(b|c)$*

We have three scenarios to consider in Bayesian Networks to understand conditional independence.

3.4.1 Tail to tail

Definition 3.4.2 *Consider the RVs a, b, c. We say that a and b are tail to tail with c iff $p(a,b,c) = p(a|c)p(b|c)p(c)$*



We know that this Bayesian network implies the following factorization

$$p(a,b) = \sum_c p(a|c)p(b|c)p(c) \neq p(a)p(b)$$

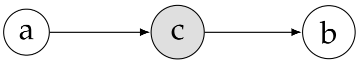
Also

$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a|c)p(b|c)p(c)}{p(c)} = p(a|c)p(b|c)$$

Which means that $a \not\perp\!\!\!\perp b$ and $a \perp\!\!\!\perp b|c$

3.4.2 Head to tail

Definition 3.4.3 *Consider the RVs a, b, c. We say that a and b are head to tail with c iff $p(a,b,c) = p(a)p(c|a)p(b|c)$ or $p(a,b,c) = p(b)p(c|b)p(a|c)$*



Again, we can use this factorization and study independence of our variables

$$p(a,b) = \sum_c p(c|a)p(b|c)p(a) = p(b|a)p(a)$$

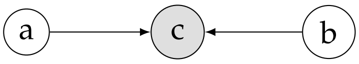
Also

$$p(a,b|c) = \frac{p(b|c)p(c|a)p(a)}{p(c)} = p(b|c)p(a|c)$$

Which means, again, that $a \not\perp\!\!\!\perp b$ and $a \perp\!\!\!\perp b|c$

3.4.3 Head to head

Definition 3.4.4 *Consider the RVs a, b, c. We say that a and b are head to head with c iff $p(a,b,c) = p(a)p(b)p(c|ab)$*



You know the drill:

$$p(a,b) = \sum_c p(c|a,b)p(b)p(a) = p(b)p(a)$$

Also

$$p(a,b|c) = \frac{p(c|a,b)p(b)p(a)}{p(c)} \neq p(b|c)p(a|c)$$

Which means, this time, that $a \perp\!\!\!\perp b$ and $a \not\perp\!\!\!\perp b|c$.
Notice that also $a \not\perp\!\!\!\perp b|d, \forall d$ descendant of c

3.4.4 Conditional independence on the graph

Let's formalize the notion of conditional independence on Bayesian Network that we just saw:

Definition 3.4.5 *Given RVs a,b,c, a path from a to b is **blocked** by c iff:*

- *c is observed and the path is head-to-tail or tail-to-tail in c.*
- *c is not observed, nor any descendant of c, and the path is head-to-head in c.*

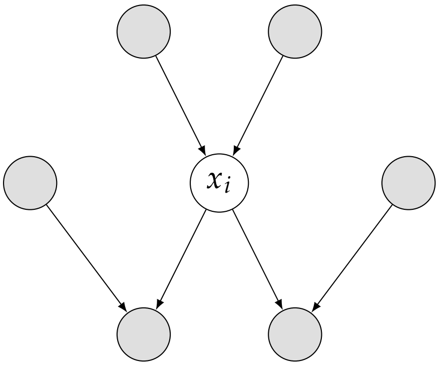
Proposition 3.4.1 *Let A, B, C subsets, if all paths from a node in A to a node in B are blocked by a node in C then $A \perp\!\!\!\perp B|C$*

3.4.5 Markov blanket

Consider a node x_i on the network and condition on everything else, i.e. on $x_{-i} = \{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\}$. Which nodes will remain in the conditioning set? If we make the computation

$$p(x_i|x_{-i}) = \frac{p(x_1, \dots, x_n)}{p(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)} = \frac{\prod_j p(x_j|pa_j)}{\sum_{x_i} \prod_j p(x_j|pa_j)}$$

In the denominator, each term in which x_i does not appear either as x_j or in $+pa_j$ will get out of the summation and cancel with the respective term in the numerator. So the only nodes that remain are those belonging to pa_i, or those for which $x_i \in pa_j$. The union of pa_i with nodes in $\{x_j\} \cup pa_j$ for which $x_i \in pa_j$ is known as the **Markov Blanket** of x_i (it contains parents, children and co-parents of x_i). Each node conditioned on its Markov blanket is independent of the rest of the network.



3.5 Naive Bayes

Naive Bayes is one of the simplest classification algorithms. Let's suppose to have a certain set of features $x_1, \dots, x_n$ dependent on a certain class c. We have the following generative model (conditioned on class c).

$$p(x_1, \dots, x_n|c)$$

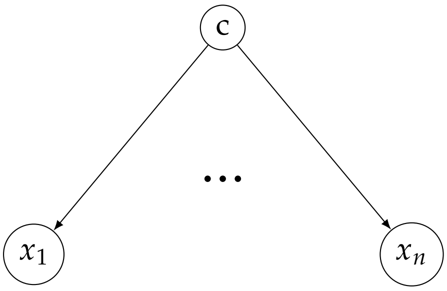
What we want to compute when we need to solve a classification task, is actually

$$p(c|x_1, \dots, x_n)$$

where $x_1, \dots, x_n$ are the features of one new data point, of which we would like to compute the probability of belonging to a certain class c. Modelling this can be very challenging.

Naive Bayes assumption We assume that our features are independent given the class c, i.e.

$$p(x_1, \dots, x_n|c) = \prod_{i=1}^{n} p(x_i|c)$$



How do we train this model? Given that we have a set of data, we consider only the x_i feature of the points of the dataset belonging to class c and fit a parametric model of $p(x_i|c;\theta)$ by means of Maximum Likelihood or other algorithms. Since we are fitting probabilistic models with one single variable, such fit is in general pretty easy.

Then, using Bayes Theorem, we know that

$$p(c|x_1,\dots,x_n) \propto p(x_1,\dots,x_n|c)p(c) = p(c)\prod_{i=1}^{n} p(x_i|c)$$

Where we also estimate $p(c)$ by taking the fraction of points belonging to class c in our dataset.

Instead of using the estimate $p(c|x_1,\dots,x_n)$, it is common practice in the context of Naive Bayes to look at the ratio

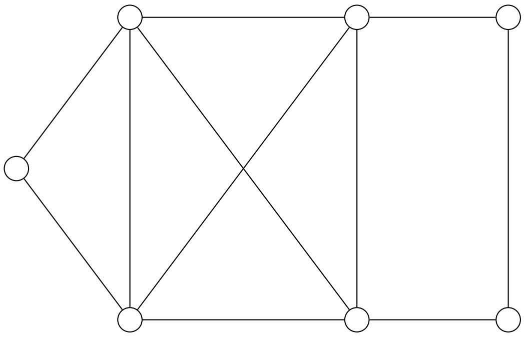
$$\frac{p(c=0|x_1,\dots,x_n)}{p(c=1|x_1,\dots,x_n)}$$

whose logarithm is known as **log-odd ratio**. Hence we can classify a certain point as belonging to class 0 if the ratio is greater than 1 (or greater than a certain classification threshold t) and belonging to class 1 otherwise. For multiclass problems, we can assign a point to the class of maximum conditional probability.

An example of application of Naive Bayes is in the domain of document classification.

3.6 Random Markov Fields

Random Markov Fields are an undirected graph representation of a probabilistic model. Nodes correspond to RV and edges to dependency. Understanding conditional independence in Markov Random Fields is easy. The way that edges model dependency, however, is more complicated.



3.6.1 Conditional independence on Markov Random Fields

Proposition 3.6.1 *Consider three subsets A, B, C. $A \perp\!\!\!\perp B|C$ iff all paths from a node $a \in A$ to any node $b \in B$ pass from C (i.e. are blocked by a node in C).*

Definition 3.6.1 *A **Markov Blanket** in a Markov Random Field for a given x_i is the set of neighbours of x_i*

3.6.2 Factorization

Consider two nodes x_i and x_j. If there is no edge from x_i to x_j we know that $x_i \perp\!\!\!\perp x_j|x_{-\{i,j\}}$. Therefore

$$p(x_i, x_j|x_{-\{i,j\}}) = p(x_i|x_{-\{i,j\}})p(x_j|x_{-\{i,j\}})$$

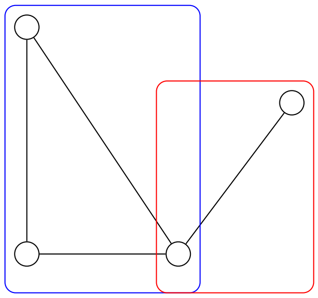
and x_i, x_j belong to different factors.

This means that nodes connected by an edge should belong to the same factor(s). Extending the reasoning, is a subgraph is fully connected, then all its nodes should be in the same factor. Therefore, graphically speaking, factors correspond to **maximal cliques** in the graph.

Definition 3.6.2 *A **clique** is a fully connected subgraph.*

Definition 3.6.3 *A **maximal clique** is a clique that cannot be extended.*

Example: The following plot highlights the two maximal cliques of an undirected graph



Let's now consider $\mathscr{C}$ = {set of maximal cliques} and $x_C = \{x | x \in C, C \in \mathscr{C}\}$

Then the factorization of a Markov Random Field is obtained as

$$p(x) = \frac{1}{Z} \prod_{c \in \mathscr{C}} \Psi_C(x_C)$$

Where Z is a normalization constant, and $\Psi_C(x_C)$ is some function evaluated on the variables that belong to the clique C.

Definition 3.6.4 $\Psi_C(x_C) \geq 0$ *is called the* ***potential function*** *and it must have a finite integral,* $\int \Psi_C(x_C) dx_C < \infty$.

Definition 3.6.5 $Z = \int \prod_C \Psi_C(x_C) dx_C < \infty$ *is the* ***Partition Function***

.

Being defined by a multidimensional integral, Z is typically hard to compute.

Therefore computing $p(x)$ might be too complicated, but notice that if we want to compute conditionals, which are ratios of marginal probabilities of $p(x)$, then Z cancels out, which makes them much easier to compute. Instead, if we want to compute marginals on few variables, we can work with the unnormalized potentials and normalize at the end. In this way we don't have to compute Z directly and we make our computations feasible.

One way to guarantee that $\Psi_C(x_C) \geq 0$ is to model $\Psi_C(x_C) = \exp(-E(x_C))$ where E is known as **energy**. This is called the **Boltzmann distribution**.

Remark: By convention, low energy corresponds to high potential.

The Boltzmann distribution is easy to work with because

$$\Psi_{C1}(x_{C1})\Psi_{C2}(x_{C2}) = \exp(-E_1(x_{C1}))\exp(-E_2(x_{C2})) = \exp(-E_1(x_{C1}) - E_2(x_{C2}))$$

Remark: There are things that one can model with Bayesian Networks that one cannot do with Markov Random Fields and viceversa.

Examples of Markov Random Fields are the Ising Model, which can also be used to denoise images, and Boltzmann Machines.