Probabilistic Graphical Models: Principles and Techniques

Notes by Jin Park

November - December 2017

Abstract

This report contains notes to *Probabilistic Graphical Models: Principles and Techniques* by Daphne Koller and Nir Friedman. It only covers one portion of the book, addressing the problem of **representation**. Some topics covered are directed and undirected networks, temporal networks, Gaussian networks, and exponential families. These notes are not complete in topics or depth, so interested readers should further purchase the book for a more rigorous representation of the material. These notes are NOT endorsed by the authors.

Contents

1 Foundations

1.1 Probability Theory

When we refer to the **confidence** of an event occuring, then we can use probability to quantify how sure we are that the event will occur. Formally, we define an *outcome space* Ω as the space of all possible outcomes and *events* as the subset of Ω. A *probability distribution* is a mapping from events to real values, and has the following three properties:

- $P(a) \geq 0$ for all $a \in S$
- $P(\Omega) = 1$
- If $a, b \in S$ and $a \cap b = \emptyset$, then $P(a \cup b) = P(a) + P(b)$

The interpretation of probability above is known as the *subjective* view of probability. It views probability as a subjective statement about an individual's belief that an event will come about. The second interpretation of probability is known as the *frequentist* view. Probability is simply the frequency of events.

1.1.1 Basic Concepts

Here's a potpourri of basic concepts in probability.

Conditional Probability
Information may change the confidence that we have of some event occurring. How do we account for this change in probability?

$$P(b|a) = \frac{P(b \cap a)}{a} \tag{1}$$

Chain Rule and Bayes Rule
From the definition of conditional probability, we immediately get the chain rule.

$$P(a_1 \cap a_2 \cap ... \cap a_k) = P(a_1)P(a_2|a_1)...P(a_k|a_{k-1}...a_1) \tag{2}$$

We also get Bayes Rule.

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)} \tag{3}$$

Random Variables
We may want to analyze an attribute (age group or symptoms). *Random variables* are a formal machinery for discussing attributes and the values of different outcomes. It is usually denoted by $X_{age=17}$ but it is actually a function $X(age = 17) \to \mathbb{R}$.
We may be given a *joint distribution* $P(X_1...X_k)$. To find the *marginal distribution*, we must sum up all the possible assignments of the other variables.

$$P(X_1) = \sum_i P(X_1, x_2^{(i)}, ...x_k^{(i)})$$

Note that random variables are simply sets of events which conforms to an attribute. This means that all rules that apply to events (conditioning, chain rule, bayes rule) apply to random variables as well.

Independence
Some information is not useful. We use *independence* as a way to describe the case when $P(a|b) = b$.

$$P \models (a \perp b) \qquad \text{if } P(a|b) = P(a) \text{ or } P(B) = 0 \tag{4}$$
$$P \models (a \perp b) \qquad \text{iff } P(a \cap b) = P(a)P(b) \tag{5}$$

Conditional Independence
Two events may be independent given certain information. Conditional independence is defined:

$$P \models (a \perp b|\gamma) \qquad \text{if } P(a \cap b|\gamma) = P(a|\gamma)P(b|\gamma) \tag{6}$$

Some independence properties include

$$(X \perp Y|Z) \Rightarrow (Y \perp X|Z) \qquad \text{Symmetry} \tag{7}$$
$$(X \perp Y, W|Z) \Rightarrow (X \perp Y|Z) \qquad \text{Decomposition} \tag{8}$$
$$(X \perp Y, W|Z) \Rightarrow (X \perp Y|Z, W) \qquad \text{Weak union} \tag{9}$$
$$(X \perp W|Z, Y) \wedge (X \perp Y|Z) \Rightarrow (X \perp Y, W|Z) \qquad \text{Contraction} \tag{10}$$

For positive distributions:

$$(X \perp Y|Z, W) \wedge (X \perp W|Z, Y) \Rightarrow (X \perp Y, W|Z) \qquad \text{Intersection} \tag{11}$$

1.1.2 Querying a Distribution

Throughout the book, we will be discussing the distributions of a subset of random variables given some information. A *probability query* consists of two parts:

- Evidence: subset E of random variable assignments in the model
- Query variables: subset Y of random variables

We will attempt to compute the *posterior probability distribution*, $P(Y|E = e)$.

We may also want to find the most probable assignment $y*$ given the information e. This is known as the *maximum a posteriori query* (MAP query) or *most probable explanation* (MLE)

$$MAP(W|e) = argmax_w P(w, e)$$

where $W = \mathbb{X} - E$, all other random variables besides E.

We may not want to find the most probable assignment for all other random variables W. Using only a subset of the variables $Y \subseteq W$ and $Z = W - Y$, we define *marginal MAP query* as

$$MAP(Y|e) = argmax_Y \sum_Z P(Y, Z|e)$$

1.1.3 Continuous Spaces

Some random variables, like blood pressure, is continuous. We use a *probability density function* to describe the distribution.

$$P(a \leq X \leq b) = \int_a^b p(x)dx$$

$$\int_{Val(X)} p(x)dx = 1$$

Two important continuous distributions are the uniform and Gaussian distributions.

$$p(x) = \begin{cases} \frac{1}{b-a} & b \leq x \leq a \\ 0 & else \end{cases} \qquad \text{Uniform} \qquad (12)$$

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(x-\mu)^2}{2\sigma^2} \qquad \text{Gaussian} \qquad (13)$$

1.2 Information Theory

Information theory is the theory of effectively coding and transmitting information. We must consider how to efficiently encode data to maximize the amount of data per channel and how to deal with noisy channels.

1.2.1 Compression and Entropy

Say that we want to send a large corpus of English text through a channel. One possible way to send it is to send it as an ASCII text. Another more efficient way is to create a dictionary of the words in the corpus and change each word in the corpus into a word index specified by the dictionary. The final way is *Huffman encoding*. The main idea of Huffman encoding is to assign variable-length codes to input characters of which lengths correspond to the frequency

of the corresponding word. The most frequent word gets the smallest character and the least frequent word gets the longest character. To create a Huffman tree, you must

1. Create a leaf node for each unique (word, frequency pair). Add these nodes into a priority queue.
2. Extract two leaf nodes with the lowest frequency.
3. Create an internal node with frequency equal to the sum of the two node frequencies. Add the first extracted node as the left child and the second extracted node as the right child.
4. Repeat step 2 and 3 until the priority queue has one node.

Is Huffman encoding the best we can do? Surprisingly, yes. The notion of entropy gives us the precise lower bound for the expected number of bits required to encode instances sampled from a large corpus. The *entropy* of a distribution over a random variable X is defined

$$H_P(X) = E_P[log\frac{1}{P(x)}] = \sum_x P(x)log\frac{1}{P(x)}$$

where we treat $0log(\frac{1}{0}) = 0$.

Another way to view entropy is as a measure of uncertainty about the value of X. Consider a game of asking yes/no questions until we pinpoint the value X. The entropy of X is the average number of questions we need to ask to get the answer. It might be tempting to draw analogies between entropy and variance. However, they are very different. Consider a bimodal distribution. Variance increases as the distance between the peaks increase, but entropy does not.

1.2.2 Conditional Entropy

Say that we want to encode values X and Y. What is the cost of encoding X if we already encoded Y? *Conditional entropy* is defined as

$$H_P(X|Y) = H_P(X,Y) - H_P(Y) - E_P[log\frac{1}{P(X|Y)}]$$

We can also find the joint distribution with the chain rule.

$$\begin{aligned} H_P(X_1...X_k) &= E[\frac{1}{P(X_1...X_k)}] \\ &= H_P(X_1) + H_P(X_2|X_1) + ... + H_P(X_k|X_1...X_k) \end{aligned}$$

We know $H_P(X|Y) \leq H_P(X)$ but by how much? In other words, how much information did Y give about X? The *mutual information* between X and Y is

$$I_P(X;Y) = H_P(X) - H_P(X|Y) = E_P[log\frac{P(X|Y)}{P(X)}]$$

Mutual information satisfies several properties:

- $0 \leq I_P(X;Y) \leq H_P(X)$
- $I_P(X;Y) = I_P(Y;X)$
- $I_P(X;Y) = 0$ iff $X \perp Y$

1.2.3 Relative Entropy and Distance between Distributions

We may want to compare two distributions. For example, we might want to approximate a distribution with a simpler one and evaluate the quality of the approximate distribution. A *distance metric* satisfies the following properties.

- Positivity: $d(P;Q) \geq 0$ and $d(P;Q) = 0$ iff $P = Q$
- Symmetry: $d(P;Q) = d(Q;P)$
- Triangle Inequality: $d(P;R) \leq d(P;Q) + d(Q;R)$

Although it is not a distance metric, relative entropy is often used to compare two distributions. Also known as the KL-divergence, we define *relative entropy* as

$$D(P||Q) = E_P[log\frac{P(X_1...X_n)}{Q(X_1...X_n)}]$$

Relative entropy does not satisfy symmetry and the triangle inequality, so it is not a distance metric. Relative entropy does, however, include conditioning

$$D(P(X|Y)||Q(X|Y)) = E_P[log\frac{P(X|Y)}{Q(X|Y)}$$

and so satisfies the chain rule

$$\begin{aligned} D(P||Q) =& D(P(X_1)||Q(X_1))+ \\ & D(P(X_2|X_1)||Q(X_2|X_1)) + ... \\ & D(P(X_n|X_1...X_{n-1})||Q(X_n|X_1...X_{n-1})) \end{aligned}$$

The relative entropy of marginal distributions is upper-bounded by the relative entropy of joint distributions.

$$D(P(X)||Q(X)) \leq D(P(X,Y)||Q(X,Y))$$

If $(X \perp Y)$ then

$$D(P(X,Y)||Q(X,Y)) = D(P(X)|Q(X)) + D(P(Y)|Q(Y))$$

Other distance metrics include

- L_1 distance: $||P - Q||_1 = \sum_x |P(x) - Q(x)|$
- L_2 distance: $||P - Q||_2 = (\sum_2 (P(x) - Q(x))^2$
- L_∞ distance: $||P - Q||_\infty = max_x|P(x) - Q(X)|$
- Variational Distance: $D_{var}(P;Q) = max_{a \in S}|P(a) - Q(a)|$

Variational distance is the maximal distance in probability for any event. It turns out that variational distance is half of L_1 norm!

Although these distance metrics are useful, relative entropy is usually more applicable to probability distributions because it follows the chain rule. Luckily, relative entropy is an upper bound for L_1 norm and consequently variational distance.

$$||P - Q||_1 \leq ((2ln2)D(P||Q))^{\frac{1}{2}}$$

2 Bayesian Networks

2.1 Exploiting Independence Properties

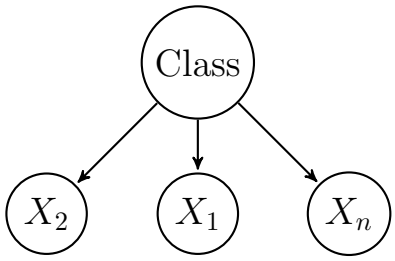
The main question of the representation problem is how to represent high dimensional distributions compactly. Using independent properties, we can reduce the amount of parameters needed to represent the distribution.

Consider a series of independent coin tosses. Assigning the result of each coin toss to random variable X_i, we would need 2^n parameters to specify the distribution; each assignment for $(X_1...X_n)$ requires a probability. However, we recognize that the probability of each coin toss is independent of each other. Then, we can specify θ_i for the probability that the coin toss will be heads. Then, $P(x_i...x_n) = \prod_i \theta_i$.

Formally, the space of all joint distributions $p_1...p_n$ is a 2^n subspace of $\mathbb{R}$, the set $\{(p_1...p_n) \in \mathbb{R}^{2^n} : p_1 + ... + p_n = 1\}$. On the other hand, the factorization of the distribution is an n-dimensional manifold in $\mathbb{R}^{2^n}$. This factorization, while being compact, does not have the same expressive power as the joint distribution.

Bayesian networks are based on the *conditional independence properties*. It will attempt to change the joint distribution $P(I, S)$ into the conditional distribution $P(I)P(S|I)$ via the chain rule.

A simple Bayesian network model is the *Naive Bayes model*. This model attempts to predict a class C based on individual features X_i.



The Naive Bayes assumption is that each feature are conditionally independent given the class: $(X_i \perp X_{-i}|C)$ for all i. With this assumption, we can factorize the distribution.

$$P(C, X_1, ..., X_n) = P(C) \prod_{i=1}^{n} P(X_i|C_i)$$

If we know each feature, we can predict the the class by maximizing the *class bias*.

$$\frac{P(C = c^1|x_1...x_n)}{P(C = c^2|x_1...x_n)} = \frac{P(C = c^1)}{P(C = c^2)} \prod \frac{P(x_i|c^1)}{P(x_i|c^2)}$$

In the case of two classes, this model uses 2n+1 parameters! Unfortunately the Naive Bayes assumption is the source of this model's flaws. In many cases, features tend to be correlated to with each other, such as symptoms for medical diagnosis or pixels in computer vision. In these cases, the Naive Bayes "overcounts" correlated features.

2.2 Bayesian Networks

At the core of the Bayesian network is a directed acyclic graph G. The graph structure can be viewed in two ways. One, it is a data structure that provides a skeleton for representing a joint distribution compactly through *conditional probability distributions* (CPDs). Second, it is a compact representation for a set of conditional independence assumptions about a distribution. We will examine both viewpoints in depth throughout the chapter.

Bayesian networks are very useful for several types of reasoning. *Causal reasoning* is the prediction of effects from causal factors. *Evidential reasoning* is the explanation of causes from effects. In evidential reasoning, there may be many causes for an event. *Intercausal reasoning* is the "explaining away" of things, where different causes of the same effect interact.

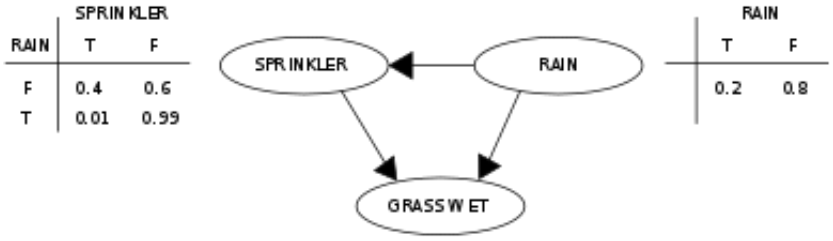
Formally, a *Bayesian network structure* is a directed acyclic graph whose nodes represent random variables $X_1...X_n$. Let Pa_{X_i} denote the parents of X_i in G and $Nondescendants_{X_i}$ denote variables that are not descendants of X_i. For each X_i

$$(X_i \perp Nondescendants_{X_i} | Pa_{X_i})$$

also called *local independencies*, $I_l(G)$. We will see in the following section that these independencies are equivalent to the conditional factorization discussed in section 2.1.

2.2.1 From Graphs to Distributions

We have defined our Bayesian network structure based only on the local independencies. In this section, we will see how this representation can be used to specify the standard way of representing a Bayesian, as a graph annotated by conditional probability distributions.



SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

The Bayesian network G is an *I-map* of probability distribution P if G associates with the set of independencies of P, $I(P)$. If G is an I-map of P, we can compress the joint representation P with the following factorization

$$P(X_1...X_n) = \prod_{i=1}^{n} P(X_i | Pa_{X_I}) \tag{14}$$

also called the *chain rule for Bayesian networks*. In a distribution of n binary random variables, we reduce the 2^n independent parameters into $n * 2^k$ parameters, where k is the maximum number of parents per node.

The above illustrates the statement that a set of independencies can construct the factorization to local conditional probability models. The converse is also true; if P is a joint distribution that factorizes according the G, aka satisfies eq. 14, G is an I-map for P. All distributions that can be factored to the chain rule of Bayesian networks can be represented by the independencies associated with the Bayesian network.

2.3 More Independencies

Independencies gives a useful framework for the task of inference, allowing us to substantially reduce the computation of a probability query. In the previous sections, we have only analyzed local independencies of Bayesian networks $I_l(G)$. What other independencies are implied by G? In other words, **given $I_l(G)$, for what subsets of random variables $X, Y, Z \in G$ can we guarantee $(X \perp Y|Z)$?**

2.3.1 D-separation

As a building block to global independencies, we'll consider four *local dependency trails* that will cause r.v. X, Y to be dependent given Z.

- Causal trail $(X \rightarrow Z \rightarrow Y)$: active iff Z is not observed
- Evidential trail $(X \leftarrow Z \leftarrow Y)$: active iff Z is not observed
- Common cause $(X \leftarrow Z \rightarrow Y)$: active iff Z is not observed
- Common effect $(X \rightarrow Z \leftarrow Y)$: active iff Z or descendant is observed. Also known as *v-structure*

A trail $X_1 \rightleftharpoons ... \rightleftharpoons X_n$ is active if it follows all of these local independencies.

Now, we can use active trails to define d-separation. Let X, Y, Z be three sets of nodes in G. X and Y are d-separated give Z if there is no active trail between any node in X, Y given Z.

$$I(G) = (X \perp Y|Z) : d - sep_G(X; Y|Z)$$

are called *global Markov independencies*. D-separation as a method of finding independencies is sound and complete.

D-separation as a way to infer independencies would be useless if there were not an efficient way for determining d-separation between two sets of random variables X, Y given Z. One way would be to check for active trails between every pair of random variable between X, Y. However, this may take an exponential amount of time depending on the graph. Fortunately there is a much faster algorithm only requiring linear time. It follows two phases

1. From leaves to roots, mark all parents of nodes in Z
2. Use breadth first search to follow local dependency trails.

2.3.2 I-Equivalence

It might be useful to analyze the similarity of one Bayesian network with another. Two graph structures K_1 and K_2 are *I-equivalent* if $I(K_1) = I(K_2)$. The set of all graphs over P is partitioned into a set of mutually exclusive and exhaustive I-equivalent classes, which are the set of equivalence classes induced by the I-equivalence relation. This means that for a given probability distribution P, there may be multiple Bayesian networks associated the distribution, $I(P) = I(G_1) = I(G_2)$. No intrinsic property allows us to favor one graph over the other. Luckily, there is a way to specify the entire class of Bayesian networks associated with P, a topic discussed further in the next section.

Which Bayesian networks are I-equivalent? If G_1 and G_2 have the same skeleton and the same set of v-structures, then they are I-equivalent. The skeleton of a Bayesian network is simply an undirected graph with edges between every pair of adjacent vertices X, Y in G. Unfortunately, this characterization is only sufficient, not complete; there exists I-equivalent graphs that do not have the same v-structure. Consider complete graphs. The global independencies of these graphs are empty, but any two may have different set of v-structures.

The reason for non-uniqueness in this example is the *covering edge*. In a v-structure, a covering edge is the edge between the parents. For $(X \rightarrow Z \leftarrow Y)$, $(X \rightarrow Y)$ or $(X \leftarrow Y)$ is a covering edge. If there is no covering edge, the v-structure is known as an *immorality*. Hence, we have our sufficient and complete condition for I-equivalence: G_1 and G_2 are I-equivalent iff they have the same skeleton and immoralities.

2.4 From Distribution to Graph

So far, we have seen the usefulness of Bayesian network as a compact representation of P and its independencies. However, for distributions in real life, P will not follow the nice factorization of Bayesian Networks. To what extent can we construct a graph G whose independencies are reasonable surrogates for the independencies in P?

One way to represent distribution P may be to take any Bayesian network that is an I-map for P. The problem with this solution is obvious; a complete graph is an I-map for every distribution, since its conditional factorization is the joint distribution. A nontrivial representation is a *minimal I-map*, an I-map for which the removal of a single edge renders it not an I-map of P. The minimal I-map can be found with the Build-Minimal-I-Map algorithm.

The main idea Build-Minimal-I-Map is simple. Let Construct a Bayesian network by adding one node at a time. Find parents of node X_i by finding the minimal set U that satisfies $(X_i \perp \{X_1 ... X_i - 1\} - U|U)$. Set these as parents because they are d-separate the current node from all other existing nodes.

Unfortunately, there are many different minimal I-maps for a particular distribution, dependent on the ordering of random variables inputted into Build-Minimal-I-Map 1. In fact, minimal I-maps may capture little or no independency at all, since it only needs to satisfy the condition that removing one node will render it not an I-Map of the distribution.

Algorithm 1 Procedure to build a minimal I-map given an ordering

procedure BUILD-MINIMAL-I-MAP($X_1 ... X_n$ an ordering of random variables, I a set of independencies.)
  Set G to an empty graph over X
  for $i = 1, ..., n$ **do**
    $a \leftarrow \{X_1, ..., X_{i-1}\}$ // U is the current candidate for parents of X_i
    for $U' \subseteq \{X_1, ..., X_{i-1}\}$ **do**
      if $U' \subset U$ and $(X_i \perp \{X_1, ..., X_{i-1}\} - U' | U') \in I$ **then**
        $U \leftarrow U'$
      end if
    end for
    // U is a minimal set satisfying $(X_i \perp \{X_1, ..., X_{i-1}\} - U | U)$
    // Now set U to be the parents of X_i
    for $X_j \in U$ **do**
      Add $X_j \rightarrow X_i$ to G
    end for
  end for
end procedure

A graph K is a *perfect I-map* (P-map) for a set of independencies of distribution P if $I(K) = I(P)$. Like a minimal I-map, there are many different P-maps for a set of independencies. To resolve this issue, we will create a partially directed acyclic graph (PDAG) that represents the all P-maps of P. Although P-maps are not unique, they are unique up to I-equivalence between networks. In the discussion in section 2.3.2, the class of I-equivalent Bayesian networks are defined by its skeleton and immoralities. We use these two components to define the PDAG $G*$ used to represent the P-map class of the distribution.

The first task is to identify the undirected skeleton of $G*$. If X and Y are two variables not adjacent in $G*$, then there exists a *witness set* U such that $(X \perp Y | U)$. The witness set U is a witness to X and Y's independence. Also note that the size of the witness set is bounded by the maximum indegree of the graph, because each node is independent of all other nodes given its parents. With this fact, we can obtain the skeleton of the distribution.

Algorithm 2 Recovering the undirected skeleton for a distribution

procedure BUILD-PMAP-SKELETON($X_1, ..., X_n$ = Set of random variables, P = distribution, d = Bound on witness set)
  Let H be the complete undirecteed graph over X
  for X_i, X_j **do**
    $U_{X_i,X_j} \leftarrow \emptyset$
    for $U \in Witnesses(X_i, X_j, H, d)$ **do**
      // Consider U as a witness set for X_i, X_j
      if $P \models (X_i \perp X_j | U)$ **then**
        $U_{X_i,X_j} \leftarrow U$
        Remove $X_i - X_j$ from H
        break
      end if
    end for
  end for
  return $(H, \{U_{X_i,X_j}\})$
end procedure

This algorithm is in $O(n^2 * \binom{n-2}{d}) = O(n^{d+2})$.

The second task is to mark immoralities within the skeleton created by Build-PMap-Skeleton. A *potential immorality* in skeleton H is three nodes $X - Z - Y$ that do not contain an edge between X and Y. There are four cases to consider, $X \rightarrow Z \rightarrow Y$, $X \leftarrow Z \leftarrow Y$, $X \leftarrow Z \rightarrow Y$, and $X \rightarrow Z \leftarrow Y$. If $G*$ indeed contains the immorality $X \rightarrow Z \leftarrow Y$, we know Z will not be contained in any witness sets $U_{X,Y}$. Also, if $G*$ contains the first three cases, we know Z will be contained in all witness sets $U_{X,Y}$. Combining these two results, **$X - Z - Y$ is an immorality iff Z is not in the witness set for X and Y.** This motivates the following algorithm.

Algorithm 3 Marking immoralities in the construction of a perfect map

procedure MARK-IMMORALITIES($X_1 ... X_n$ = the set of random variables, S = skeleton, U_{X_i,X_j} = witnesses found by Build-PMap-Skeleton)
  $K \leftarrow S$
  for X_i, X_j, X_k such that $X_i - X_j - X_k \in S$ and $X_i - X_k \notin S$ **do**
    if $X_j \notin U_{X_i,X_j}$ **then**
      Add the orientations $X_i \rightarrow X_j$ and $X_j \leftarrow X_k$ to K
    end if
  end for
end procedure

The final task is to resolve any inconsistencies with the graph. The first rule (R1) is the case when $(X \rightarrow Y - Z)$. The undirected edge $Y - Z$ must be oriented right, or it would be an immorality. The second rule (R2) is derived from the acyclicity constraint. If $(X \rightarrow Y \rightarrow Z)$ and $X - Z$, then $X - Z$ must be oriented toward the right to not create a cycle. The final rule (R3) is more complex, but uses acyclic and immorality constraints.

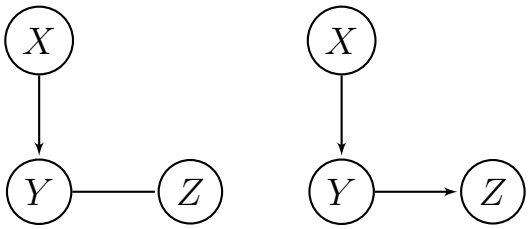


Figure 1: R1

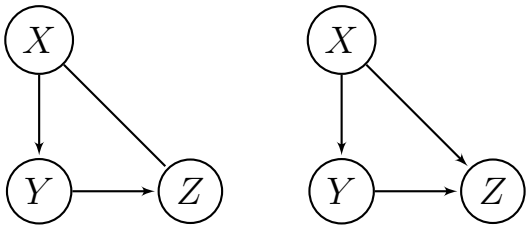


Figure 2: R2

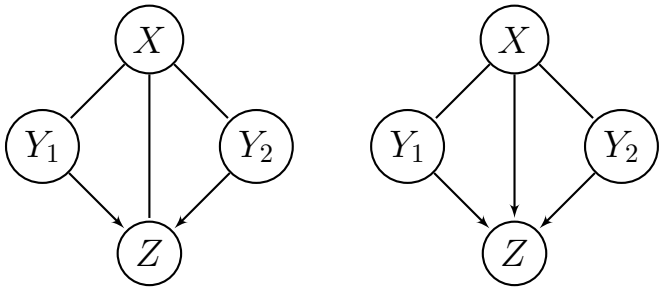


Figure 3: R3

Using these three rules, we can propagate constraints through our skeleton oriented by Mark-Immoralities. The algorithm is implemented as follows.

Algorithm 4 Finding the class PDAG characterizing the P-map of distribution

procedure BUILD-PDAG($X_1...X_n$ = the set of random variables, P = distribution)
  $S, \{U_{X_i,X_j}\} \leftarrow$ Build-PMap-Skeleton($X_1...X_n, P$)
  $K \leftarrow$ Find-Immoralites($X_1...X_n, S, \{U_{X_i,X_j}\}$)
  while not converged **do**
    Find a subgraph in K matching the left-hand side of rules R1-R3
    Replace the subgraph with the right-hand side of the rule
  end while
  return K
end procedure

This algorithm is sound and complete for all distributions P that have a perfect map.

3 Undirected Graphical Models

There are some probabilistic distributions that cannot be captured with Bayesian Networks. Consider the following. There are four students Alice, Betty, Carl, and Debbie, who are collaborating in homework with each other. However, Alice refuses to work with Carl, and Betty will not work with Debbie. To find the probability that all four students receive full credit on their homework, the distribution $P(A, B, C, D)$ satisfies only the independencies $(A \perp C|B, D)$ and $(B \perp D|A, C)$. Any Bayesian network I-map would have extraneous edges, and so would not capture one of the independencies.

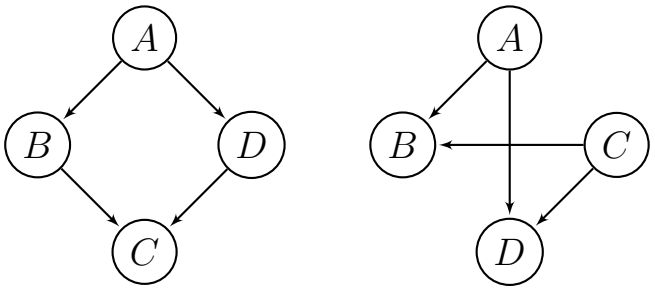


Figure 4: Attempt at a Bayesian Network

The first graph includes the extra independency $(B \perp D|A)$. The second graph also includes an extra independency $(A \perp C)$.

A more natural representation is to have undirected edges. As you can see below, an undirected graph perfectly captures the set of independencies. This graphical model is called a *Markov network*, a network in which nodes are variables, and edges are "probabilistic relationships".

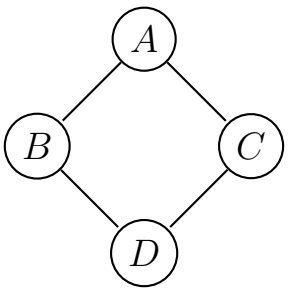


Figure 5: Markov Network

In a Bayesian network, we used CPDs to weight edges. However, in a Markov network, there is no direction. We instead use *factors*; a factor ϕ is a function from a set of random variables $D = \{X_1...X_k\}$ to $\mathbb{R}$. D is known as the *scope* and is also denoted $Scope[\phi]$.

3.1 Parameterization

It might be tempting to associate factors directly to edges in a graph. However, this limits the expressive power of the model. Consider a fully connected Markov network over P. If all variables are binary, each factor over each edge has 4 parameters, so the total number of parameters would be $4\binom{n}{2}$. However, P has no independencies and therefore specifies a fully joint distribution, requiring $2^n - 1$ parameters. Associating factors directly to edges only encodes pairwise relationships. A more general representation is to allow factors over arbitrary number of variables.

We may want to specify the fully joint distribution of a Markov network. A *Gibbs distribution* parameterized by a set of factors $\Phi = \{\phi_1(D_1), ..., \phi_k(D_k)\}$ if

$$P_\Phi(X_1...X_n) = \frac{1}{Z}\widetilde{P}_\Phi(X_1...X_n)$$

where

$$\widetilde{P}_\Phi = \phi_1(D_1) * ... * \phi_k(D_k) \qquad \textit{(unnormalized measure)}$$
$$Z = \sum_{X_1...X_n} \widetilde{P}_\Phi(X_1...X_n) \qquad \textit{(partition function)}$$

Each D_k is called a clique potential.

We may want to condition our Markov network to a certain context U. Say that we have some new information $U \subset Y$ and factor $\phi(Y)$. A *factor reduction* of ϕ to the context $U = u$ is a factor over scope $Y' = Y - U$ such that

$$\phi[u](y') = \phi(y', u)$$

This simply removes all entries within the original factor $\phi(y)$ that is inconsistent with u. A *reduced Gibbs distribution* $P_\Phi[u]$ to the context u is a Gibbs distribution defined by the set of factors $\Phi[u] = \{\phi_1[u], ..., \phi_k[u]\}$. Note that $P_\Phi[u] = P_\Phi(W|u)$ where $W = X - U$.

Let H be a Markov network over X and context $U = u$. A *reduced Markov network* $H[u]$ is a Markov network over the nodes $W = X - U$, where we have an edge $X - Y$ if there is an edge $X - Y$ in H. Note that this is the same operation as reducing the Gibbs distribution but in graphical form.

3.2 Markov Network Independencies

Like the Bayesian network, we must ensure that Markov networks can encode a set of independencies. Intuitively, separation between X, Y is when influence cannot "flow" between one node and the other. An *active path* is a path $X_1 - ... - X_n$ where there is no observed variable Z that blocks it. *Separation* occurs when there is no active path between two subsets of random variables X, Y. We define *globalindependencies* encoded by Markov network H as

$$I(H) = \{(X \perp Y|Z) : sep_H(X; Y|Z)\}$$

Separation is monotonic as Z increases. That is, observing more variables cannot induce an active path.

With these definitions, we can prove that a Gibbs distribution P over graph H follows the independencies in I-map H, $I(H)$. Proving the sufficiency of this

equivalence is left for exercise. The completeness only holds if P is a positive distribution and is known as the Hammersly-Clifford theorem.

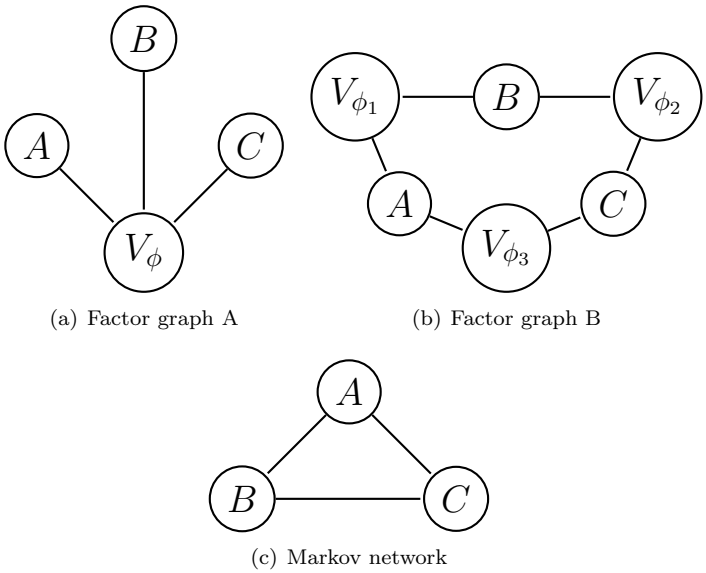
There are three types of Markov independencies.

- Global: $I(H) = \{(X \perp Y|Z) : sep_H(X;Y|Z)\}$
- Pairwise: $I_p(H) = \{(X \perp Y|\mathbb{X} - X, Y) : X - Y \notin H\}$
- Local (Markov blanket): $I_l(H) = \{(X \perp \mathbb{X} - X - MB_H(X))|MB_H(X)) : X \in \mathbb{X}\}$

The set of global independencies will always be larger than the set of local independencies, and the set of local independencies will be larger than pairwise independencies, $I(H) \supseteq I_l(H) \supseteq I_p(H)$. For positive distributions, all of them are equal, $I(H) = I_l(H) = I_p(H)$.

3.3 Parameterization Revisited

A Markov network does not generally reveal all the stucture in a Gibbs parameterization. In a complete subgraph, we cannot tell whether the factors are pairwise or over different subsets of the clique.



(a) Factor graph A

(b) Factor graph B

(c) Markov network

Figure 6: Different Factor graphs for the same Markov Network

A *factor graph* F is an undirected graph with variable nodes and factor nodes V_ϕ. A distribution factorizes over F if it can be represented as the set of factors represented by V_ϕ.

3.3.1 Log-linear models

Sometimes patterns that arise between different variables can be easier seen in log-space. An *energy function* is a factor in log-space

$$\epsilon(D) = -ln(\phi(D))$$

The energy function is a *feature*, a function from $Val(D)$ to $\mathbb{R}$, where D is the scope. A feature is analogous to a factor without nonnegativitivity constraints. We can rewrite the Gibbs distribution in feature form, giving us the log-linear model.

$$P(X_1...X_n) = \frac{1}{Z} exp[-\sum_{i=1}^{k} w_i \epsilon_i(D_i)]$$

There are three types of representation of Markov networks.

1. product over clique potentials (Markov factorization)
2. product over factors (factor graph)
3. product over feature weights (log-linear model)

Each factorization is more fine-grained than the previous; that is, a factor graph can model all Markov factorizations, and a log-linear model can model all factor graphs. All representations are useful. Markov models are useful for analyzing independence assertions. Factor graphs are useful for inference, and features are useful for parameterization in learning as they provide the biggest capacity.

An application of log-linear models are the Boltzman distribution. The Boltzman distribution is driven by the idea that a neuron is activated by the strength of the signal coming from neighboring neurons. The probability of X_i is

$$P(X_i) = sigmoid(-\sum_{j}(w_{i,j}x_j) - w_i)$$

where w_i is the weight of the connection between X_i and its neighbor X_j, x_j is the activation of neighbor X_j, and w_i is some bias on neuron X_i. This the most popular mathematical approximation of the function employed by a neuron in the brain. Thus, if we imagine a process by which the network continuously adapts its assignment by resampling the value of each variable as a stochastic function of its neighbors, then the "activation" probability of each variable resembles a neuron's activity. This model is a very simple variant of a stochastic, recurrent neural network.

3.3.2 Eliminating Ambiguity

Unfortunately there are infinitely ways of parameterizing a log-linear model. Consider a distribution $P(A, B, C)$ such that there are two energy functions $\epsilon_1(A, B)$ and $\epsilon_2(B, C)$. For any constant λ, we can redefine

$$\epsilon_1'(a, b^i) := \epsilon_1(a, b^i) + \lambda$$
$$\epsilon_2'(b^i, c) := \epsilon_2(b^2, c) + \lambda$$

and these new energy functions would be valid parameterizations of P.

The *canonical parameterization* of a Gibbs distribution resolves this ambiguity. Let x_Z be the assignment in x to Z, and ξ_{-Z} be the assignment of all other variables outside Z. Then, the *canonical energy function* is

$$\epsilon_D^*(d) = \sum_{Z \subseteq D} (-1)^{|D-Z|} l(d_Z, \xi_{-Z}^*)$$

and the *canonical parameterization* of the Gibbs distribution is

$$P(\xi) = exp[\sum_i \epsilon^*_{D_i}(\xi(D_i))]$$

3.4 From Bayesian networks to Markov networks

The factorization of Bayesian networks can be reduced to a Gibbs distribution.

$$\begin{aligned} P(X_1...X_n) &= \prod_i P(X_i|Pa_{X_i}) \\ &= \prod_i \phi_{X_i}(X_i, Pa_{X_i}) \end{aligned}$$

A Bayesian network conditioned on evidence $E = e$ induces a Gibbs distribution reduced to the context $E = e$.

To create a Markov graph structure from a Bayesian network G, we can moralize the skeleton of G. A *moralized graph* $M(G)$ of Bayesian network G is an undirected graph that contains an edge $X - Y$ if: 1. X and Y share an edge in G or 2. X and Y are parents of the same node. The term "moralize" comes from the fact that two parents within an "immorality" will be "married". The moralized graph is guaranteed to be a minimal I-map for G. If G contains no immoralities then $M(G)$ is a perfect map for G.

3.5 Partially Directed Models

In this section, we will unify directed and undirected graphical models. This way, we are able to express both directed and undirected independencies.

A *conditional random field* is an undirected graph H whose nodes correspond to $X \cup Y$. The network encodes a conditional distribution

$$\begin{aligned} P(Y|X) &= \frac{1}{Z(X)}\widetilde{P}(Y, X) \\ \widetilde{P}(Y, X) &= \prod_{i=1}^{m} \phi_i(D_i) \\ Z(X) &= \sum_Y \widetilde{P}(Y, X) \end{aligned}$$

such that each $D_i \not\supseteq X$.

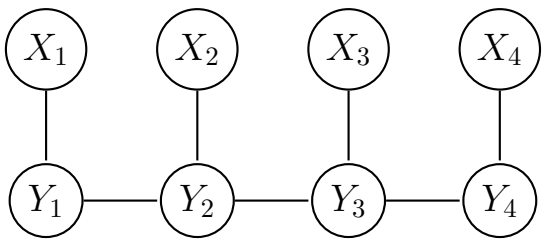


Figure 7: Conditional Random Field

We avoid encoding the distribution over variables in X. This allows the model a rich set of observed variables X. Defining a conditional distribution lets the observed variables be more expressive!

A *chain graph* is a partially acyclic directed graph such that each chain component is a conditional random field on its parent's chain component. The moralized graph corresponds to fully connecting parent components together. Let $P(K_i|Pa_{K_i})$ be a CRF over the parents and K_i, then we can factorize the distribution P over the chain graph

$$P(X_1...X_n) = \prod_{i=1}^{l} P(K_i|Pa_{K_i})$$

We can define all the different independencies in PDAG K.

- Pairwaise: $I_p(K) = \{(X \perp Y|(Nondescendants_X - X - Y) : X, Y \text{ is nonadjacent}, Y \in Nondescendants_X\}$
- Local: $I_l(K) = \{(X \perp Nondesc_X - Boundary_X|Boundary_X)\}$
- Global: $(X \perp Y|Z)$ if X is separated from Y in $M[K[X \cup Y \cup Z]]$ given Z

All positive distributions P factorizes over PDAG K iff $P \models I(K)$.

4 Local Probabilistic Models

So far, we have discussed how to model probability distributions into Bayesian networks and Markov networks. Conditional probability distributions (CPDs) represent the relationship between each node with its parents in a Bayesian network, $P(X|Pa_X)$. How many different ways are there to model CPDs?

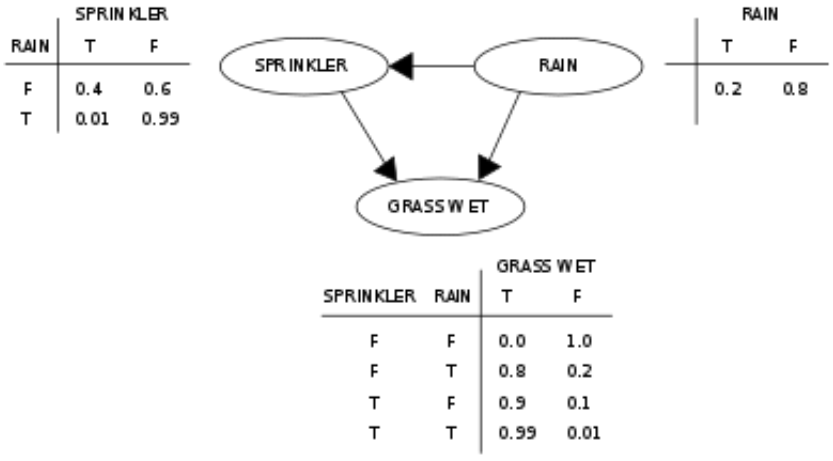
4.1 Tabular CPDs

For a Bayesian network with random variables of discrete values, we can use a table to represent all $P(X|Pa_X)$ explicitly. Each entry in a *tabular CPD* represents a set of the $P(x|pa_X)$ for a particular assignment for the node x and parents pa_X. It must satisfy:

- Non-negativity: $P(x|pa_X)$
- Normalized: $\sum_{x \in Val(X)} P(x|pa_X) = 1$

Unfortunately, the tabular representation of CPDs have several key disadvantages. It can only store discrete joint distributions and cannot encode to an infinite domain. The tabular representation also grows exponentially by the number of parents, as the number of entries is $|Val(Pa_X)| * |Val(X)|$.

Another major flaw is that a table cannot explicitly represent the structure between Pa_X and X. Solution? Represent the CPD as a function that, given pa_X and x, returns $P(x|Pa_x)$.



4.2 Deterministic CPDs

A *deterministic function* maps a value of parents Pa_X to a value of X:

$$P(x|pa_X) = \begin{cases} 1 & x = f(pa_X) \\ 0 & \text{else} \end{cases}$$

For example, X might be the "or" of its parents. Or $P(X|Y,Z)$ might imply that $X = Y + Z$.

When modeling a car, we might have four variables $T_1, ..., T_4$, each corresponding to a flat in one of the four tires. Naively, we can make each T_i a parent of all affected variables, like *Steering* or *Ride*. However, we can make an intermediate variable $Flat - tire$ be the OR of all T_is, significantly reducing the number of parameters required for the CPDs.

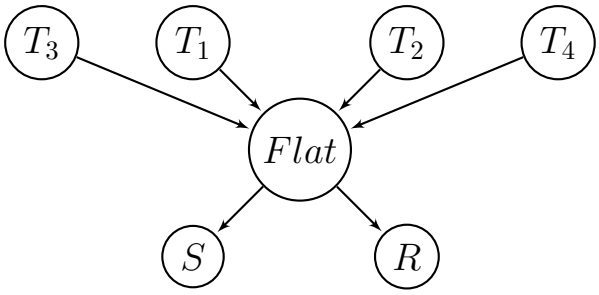


Figure 8: Flat tire scenario

Deterministic functions can augment independencies. Consider the above example. T_is determine $Flat$. Therefore, $(S \perp R|T_1...T_4)$. Note that this is not explicit in the global independencies of Bayesian networks.

We can create an algorithm to exploit the fact that observed variables may add more observed random variables through deterministic functions. The main idea is to add these additionally observed variables into an expanded observed set Z^+.

Algorithm 5 Computing d-seperation in the presence of deterministic CPDs

procedure DET-SEP($Graph$ = network structure, D = set of deterministic variables, X, Y, Z = query)
  $Z^+ \leftarrow Z$
  while there is an X_i that $X_i \in D$ and $Pa_{X_i} \subseteq Z^+$ **do**
    $Z^+ \leftarrow Z^+ \cup \{X_i\}$
  end while
  return d-sep$(X; Y|Z^+)$
end procedure

This algorithm is sound and finds all independencies implied by the graph structure and set of deterministic CPDs D.

Some deterministic functions can induce additional independencies. In the flat-tire scenario, having a flat in one tire T_1s will cause steering and ride to be affected, regardless of the status of the other tires. In other words, $(S, R \perp T_2, T_3, T_4|t_1^1)$. Deterministic functions like OR can imply a type of independence that only holds for particular values.

Let X, Y, Z be a pairwise disjoint set of variables and C be a set of variables not necessarily disjoint. We say X and Y are *contextually independent* given Z and c if

$$P(X|Y, Z, c) = P(X|Z, c) \qquad \text{whenever } P(Y, Z, c) > 0$$

These independences are called *context-specific independencies* (CSI).

4.3 Context-specific CPDs

We can make context-specific independencies explicit. Consider the following graph.

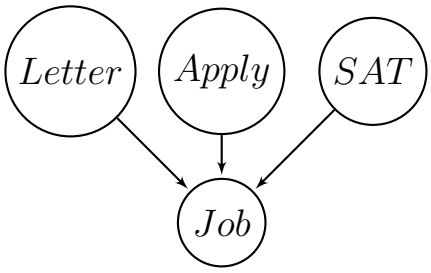


Figure 9: Job Application Scenario

Say we want to find the CPD $P(J|A,S,L)$. If teh student does not apply ($A = a^0$), there is a chance that the company might recruit the student. However, they won't have access to recommendation letters or SAT scores. This means among the 8 values of parents A, S, L, four with $A = a^0$ must induce an identical distribution. How do we represent this regularity within the CPD function?

4.3.1 Tree-CPDs

A *tree-CPD* is a rooted tree representing a CPD. Each leaf is labeled with a distribution $P(x)$ and interior node is labeled with some variable $Z \in Pa_X$. Each interior node has outgoing edges to its children, associated with a variable assignment $Z = z^i$. A *branch* is a path beginning at the root to a leaf node, following the context induced by the branch.

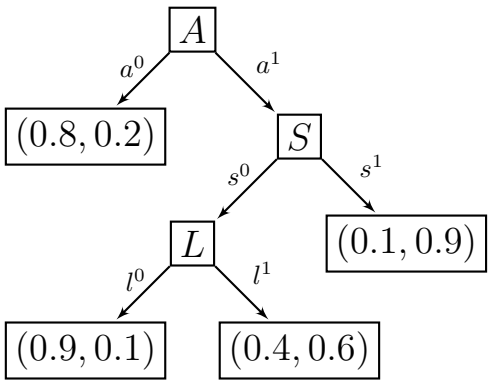


Figure 10: Tree-CPD for Job Application Scenario

We can see that the tree models the structural relationships of $P(J|A,S,L)$ described above. This representation also requires less parameters. We need parameters for every node of the tree, since those capture all possible contexts. In the Job application example, we only need 4 parameters, as opposed to the 8 parameters required by a tabular-CPD.

Trees are useful for context-specificity. It's easy to interpret and also easy to learn automatically from a dataset.

4.3.2 Rule-based CPD

A finer-grained representation of CPDs is rules. A rule ρ is a pair $< c; p >$ where c is an assignment to some subset of variables C, and $p \in [0, 1]$. C is the *scope* of ρ, denoted $Scope[\rho]$. Rules decompose trees into its basic elements. Consider the following rule-CPD of $P(J|A, S, L)$.

$$\begin{aligned}
\rho_1 &:< a^0, j^0; 0.8 > \\
\rho_2 &:< a^0, j^1; 0.2 > \\
\rho_3 &:< a^1, s^0, l^0, j^0; 0.9 > \\
\rho_4 &:< a^1, s^0, l^0, j^1; 0.1 > \\
\rho_5 &:< a^1, s^0, l^1, j^0; 0.4 > \\
\rho_6 &:< a^1, s^0, l^1, j^1; 0.6 > \\
\rho_7 &:< a^1, s^1, j^0; 0.1 > \\
\rho_8 &:< a^1, s^1, j^1; 0.9 >
\end{aligned}$$

In this case, each rule corresponds to a branch in the tree-CPD.

A *rule-based CPD* $P(X|Pa_X)$ is a set of rules R such that

- $Scope[\rho] \subseteq X \cup Pa_X$
- Each assignment has one rule
- The resulting CPD is valid (non-negative and normalized).

We can compute entire joint distributions of Bayesian networks by multiplying a set of rule-CPDs together. Let Ξ be a full instantiation of a BN and R be a multiset of R_X containing rules for $P(X|Pa_X)$. Then,

$$P(\xi) = \prod_{\rho \in R, \xi \sim c} \rho$$

This simply multiplies all rules consistent with the global context ξ. We can easily prove this fact from the factorization of Bayesian networks.

4.3.3 Other representations

Other representations include decision trees, multinets, and similarity networks.

- Decision trees - tree-CPDs that allows shared children. This prevents repeated subtrees.
- Multinet - Multiple Bayesian networks B_c dependent on the context $C = c^i$. A multinet can be many layers.
- Similarity network - multiple Bayesian networks dependent on a subset of attributes. A multinet is dependent on variable assignments, while a similarity network is dependent on related attributes.

4.3.4 Independencies

We apply the concept of conditioning to rules. Say c is a context and c' is a rule compatible with c. A *reduced rule set* is

$$R[c] = \{\rho[c] =< c', p >: \rho \in R, \rho \sim c\}$$

We can define context-specific independencies by finding all $Y \subseteq Pa_x$ such thate Y is not the scope of the rules in $R[c]$. In other words,

$$(X \perp_c Y | Pa_X - Y, c)$$

In Bayesian networks, an edge $X \to Y$ in context c is *spurious* if it satisfies the above independency. The intuition behind spurious edges is that they are removed under certain contexts defined by context-specific independencies.

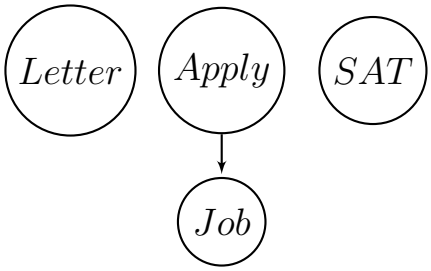


Figure 11: Job Application Scenario when not applied

In the Job Application example, remember that probability of getting an offer is not dependent on letters or SAT scores if the applicant did not apply. The $L \to J$ and $S \to J$ is a spurious edge in context $A = a^0$.

There is an algorithm to integrate spurious edges into d-seperation.

Algorithm 6 Computing d-seperation in the presence of context-specific CPDs

```
procedure CSI-SEP(G = Bayesian network, c = context, X, Y, Z = query)
    G' ← G
    for each edge Y → X in G' do
        if Y → X is spurious given c in G' then
            Remove Y → X in G'
        end if
    end for
end procedure
```

CSI-separation is a variant of d-separation induced by certain context $C = c$. It is formally defined by the algorithm above. CSI-separation as a criterion for context-specific independencies is sound, but not complete.

Consider another job application example that only requires two recommendation letters. However, due to an interviewer bottleneck, only one letter can be read.

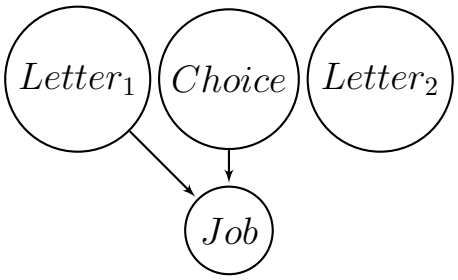


Figure 12: Job Application Scenario 2 when the first letter is chosen

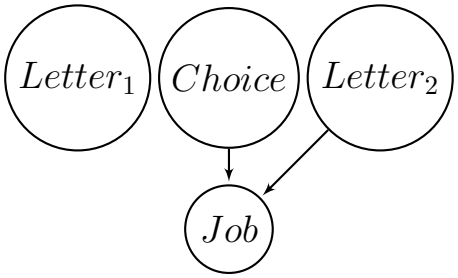


Figure 13: Job Application Scenario 2 when the second letter is chosen

Under the first context c^1, when the first letter is chosen, CSI-sep will find that $L_2 \rightarrow J$ is spurious, so $(L_1 \perp L_2 | J, c^1)$. Under the second context c^2, when the second letter is chosen, CSI-sep will find that $L_1 \rightarrow J$ is spurious, so $(L_2 \perp L_1 | J, c^2)$. However, under an empty context, CSI-sep finds no spurious edges and does not find the context-specific independency $(L_1 \perp L_2 | J, C)$. We can see through this example that CSI-sep cannot find all context-specific independencies because it cannot reason by cases.

4.4 Independence of Causal Influence

In many cases, each parent of a node simply contributes some influence on X. How can we model $P(X|Pa_X)$ as simply a combination of influences?

4.4.1 The Noisy-Or Model

Say the probability that a student will get a good grade is dependent on two binary variables: asking good questions Q and the final paper P. However, even if he/she asks a good question, the professor might forget it was him/her. To make matters worse, even if he/she writes a good paper, the professor might have a hard time reading the handwriting. We can model this by introducing two new variables: asking good questions and professor remembering Q' and writing a good paper and professor being able to read the handwriting P'. The relationship can be modeled

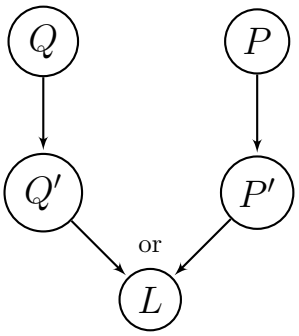


Figure 14: Noisy-Or model for Letter Grade Example

We can generalize the noisy-or model. $P(Y|X_1...X_k)$ is *noisy-or* if there are $k+1$ noise parameters $\lambda_0, ..., \lambda_k$ such that

$$P(y^0|x_1...x_k) = (1-\lambda_0)\prod_{i=1}^{k}(1-\lambda_i)^{x_i}$$

Each λ_i is called a *leak variable*.

A useful property of noisy-or models is that each parent X_i is independent of each other given the child Y. In other words,

$$(X_i \perp X_j|Y) \qquad \text{for all i,j}$$

An application of noisy-or models is BN2O networks. These two-layer networks create a decent baseline implementation for medical diagnosis. Each disease connects to every symptom, and each symptom's CPD $P(S_i|D_1...D_k)$ is a noisy-or model. During inference, BN2O networks take advantage of the fact that each disease is independent of each other given the symptoms.

4.4.2 Generalized Linear Models

Consider the body's immune system. Each invader adds to the burden the body's immune system has to bear until its defenses cannot hold anymore. We can model the probability of a fever as a linear function passed through a smooth thresholding function

$$P(Y|X_1...X_k) = \sigma(w_0 + \sum_{i=1}^{k} w_i X_i)$$

where σ is the sigmoid function. This linear model is called a *logistic CPD* and is equivalent to a naive Markov model.

The above only applies to binary random variables. We can easily extend the CPD to the multiple-values by essentially one-hot encoding the multiple values of X_i and Y and passing them through the softmax function. The CPD

$P(Y|X_1...X_k)$ is a *mulinomial logistic* if

$$l_j(X_1...X_j) = w_{j,0} + \sum_{i=1}^{k} w_{j,i}X_i$$

$$P(y^j|X_1...X_k) = \frac{exp(l_j(X_1...X_k))}{\sum_{j'=1}^{m} exp(l_{j'}(X_1...X_k))}$$

4.4.3 General Formulation

Both of the described models satisfy a property called the *independence of causal influence* (ICI). Formally, the CPD $P(Y|X_1...X_k)$ exhibits ICI if it can be described as the following pairwise conditional random field.

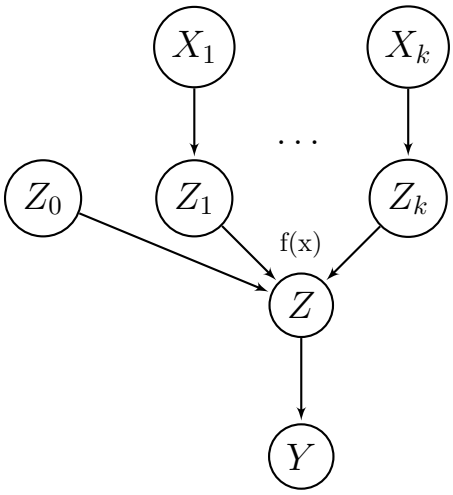


Figure 15: Independence of causal influence

Intuitively, each X_i indicates a hidden binary variable Z_i that deterministically contributes to a threshold Z that ultimately determines Y.

This definition isn't particularly useful because $f(x)$ can be an arbitrarily complex function. A more useful definition follows from symmetric decomposability. A function $f(x_1...x_k)$ is *symmetric decomposable* if there exists a communicative associative function $x \diamond y$ such that $f(x_1...x_k) = x_1 \diamond ... \diamond x_k$. A CPD $P(Y|X_1...X_k)$ exhibits *symmetric ICI* if it can be described by the above network and the CPD of Z is a deterministic symmeteric decomposable function. The CPD exhibits *fully symmetric ICI* if the CPDs of different Z_is are identical.

4.5 Continuous Variables

So far, we have only considered discrete variables of finite values. Some variables, however, are modeled best in continuous space. How can we model the CPD $P(X|Pa_X)$ of continuous random variables?

One approach is to discretize intervals. If we have a continuous random variable for position P, we can create a discrete variable P' that tells us whether

an object is in $[0cm, 15cm], [15cm, 30cm], ...$ The issue with this is that the number of parameters would explode. Each random variable X will require $\frac{\text{size of domain}}{\text{length of an interval}}$ and the tabular CPD will require an exponential number of this factor. Another issue is that you cannot capture an infinite domain. Lastly, you lose the structure of the distribution. In the position example, there is no notion of "closeness" between two values of P'. The value $p'^1 = [0cm, 15cm]$ is the same distance to $p'^2 = [15cm, 30cm]$ as it is to $p'^3 = [30cm, 45cm]$.

The second approach is to create a linear Gaussian CPD. Let Y be a continuous variable with continuous parents $X_1, ..., X_k$. Y has a *linear Gaussian model* if

$$p(Y|x_1...x_k) = \mathcal{N}(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k; \sigma^2)$$

We can also write it as a linear function of random variables

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_k X_k + \epsilon$$

where $\epsilon \sim \mathcal{N}(0; \sigma^2)$. Unfortunately, this model doesn't allow the variance of Y to be dependent on its parents X_i.

The last approach is to use several linear Gaussian CPDs to incorporate both discrete and continuous random variables.

4.5.1 Hybrid Models

Say we want to incorporate both discrete and continuous random variables. There are two cases we must consider: discrete/continuous parents determining a continuous child and discrete/continuous parents determining a discrete child.

For the first case, we can simply define a linear Gaussian CPD for every entry in the table-CPD generated by the discrete parents. Let X be a continuous variable, $U = \{U_1...U_m\}$ be its discrete parents and $Y = \{Y_1...Y_k\}$ be its continuous parents. X has a conditional linear Gaussian (CLG) if for all $u \in Val(U)$

$$P(X|u, y) = \mathcal{N}(a_{u,0} + \sum_{i=1}^{k} a_{u,i} y_i; \sigma_u^2)$$

A CLG network is a BN in which discrete variables only have discrete parents and continuous variables has a CLG CPD.

A CLG induces a *Gaussian mixture*, a weighted average of Gaussians. Since each Gaussian component corresponds to a context u, the number of Gaussain components is exponential to the number of discrete parents.

For the second case, we can use a threshold function of some form

$$P(u') = \begin{cases} 0.9 & y \leq 65 \\ 0.05 & else \end{cases}$$

The issue with this representation is that it is coarse and discontinuous. A more natural way to represent discrete random variables is to use a logistic or softmax function. Of course, this will create random variables that are continuous but act similar to discrete variables.

4.6 Conditional Bayesian network

The pairwise CRF representing the independence of causal influence 15 does not specify a joint distribution over the random variables X and Y. Rather, it describes the conditional distribution between parents X and child Y through hidden variables Z. We can generalize this concept to any Bayesian network.

A *conditional Bayesian network* has nodes in disjoint sets of random variables $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ The variables in $\mathbb{X}$ are called inputs, $\mathbb{Y}$ is called the outputs, and $\mathbb{Z}$ is encapsulated. $X \in \mathbb{X}$ has no parents. The network defines a conditional distribution

$$P_\beta(\mathbb{Y}, \mathbb{Z}|\mathbb{X}) = \prod_{X \in \mathbb{Y}, \mathbb{Z}} P(X|Pa_X)$$

The marginal distribution $P(\mathbb{Y}|\mathbb{X})$ is computed

$$P_\beta(\mathbb{Y}|\mathbb{X}) - \sum_{\mathbb{Z}} P_\beta(\mathbb{Y}, \mathbb{Z}|\mathbb{X})$$

An *encapsulated* CPD is one represented using a conditional Bayesian network. Encapsulation black-boxes the individual CPDs and is useful from an engineering perspective for the following reasons

- Complex systems can be built with encapsulated subsystems
- The are easier to debug
- Similar subsystems can us the same Bayesian network templates via Interfaces and subclasses

These advantages creates a framework for object-oriented Bayesian networks.

5 Template-Based Representations

So far, the models we dealt with were variable-based, where the set of random variables and relationships between variables were fixed. Only the values of the variables could change. In this section, we introduce a higher-level representation that allows us to scaffold more expressive models through templates.

5.1 Temporal Models

We will begin by observing a specific type of template model: the temporal model. *Temporal models* are used to model dynamic settings, where the state of the world evolves over time t. $X_i^{(t)}$ represents an instantiation of the template variable X_i. A *trajectory* is a "possible world" in our probability space: an assignment of values for each variable $X_i^{(t)}$ for each relevant time t.

We assume that our system is a set of time slots: measurements of the system state taken at predetermined intervals. Since the current state $\mathcal{X}^T$ is only dependent on attributes in previous time steps, we can use the chain rule to reparameterize the distribution

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)}|\mathcal{X}^{(0:t)})$$

If we assume the Markov assumption $(\mathcal{X}^{(t+1)} \perp \mathcal{X}^{(0:(t+1))}|\mathcal{X}^{(t)})$

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{(T-1)} P(\mathcal{X}^{(t+1)}|\mathcal{X}^{(t)})$$

Graphically, the markov assumption implies variables at state t is only directly connected to variables at state $t-1$.

In practice, the Markov assumption needs only to be a reasonable approximation to the world. In most cases, we can apply this assumption. We can also define models that are semi-Markov.

W can assume a dynamic system is *stationary* where $P(\mathcal{X}^{(t+1)}|\mathcal{X}^{(t)})$ is the same for all t. We represent a *transitional model* $P(\mathcal{X}^{t'}|\sqcup)$ with

$$P(\mathcal{X}^{(t+1)} = \xi'|\mathcal{X}^{(t)} = \xi) = P(\mathcal{X}'|\mathcal{X})$$

5.1.1 Dynamic BN

We can represent probability distributions over infinite trajectories compactly with the initial distribution and transition model $P(\mathcal{X}'|\mathcal{X})$. The transition model is a type of conditional BN called a *2-time-slice BN* (2-TBN). A 2-TBN for a process over $\mathcal{X}$ is a conditional Bayesian Net over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I$ is a set of interface variables.

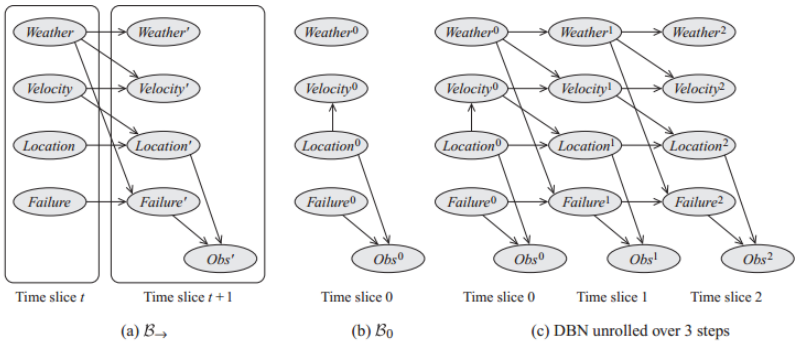
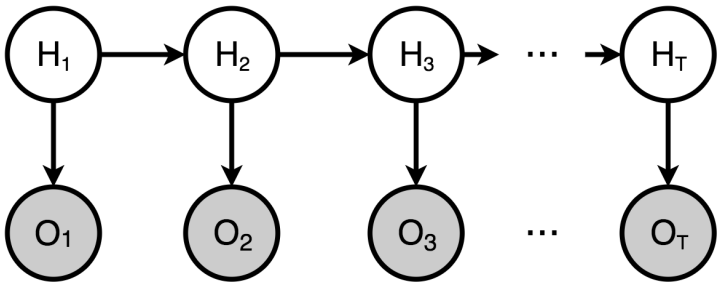


Figure 6.1 A highly simplified DBN for monitoring a vehicle: (a) the 2-TBN; (b) the time 0 network; (c) resulting unrolled DBN over three time slices.

For each template variable X_i, the CPD $P(X_i'|Pa_{X_i'})$ is a *template factor*. A *dynamic Bayesian network* is a pair $(\beta_0, \beta_{\rightarrow})$ where β_0 is the initial state and $\beta_{\rightarrow}$ is a 2-TBN for the process.

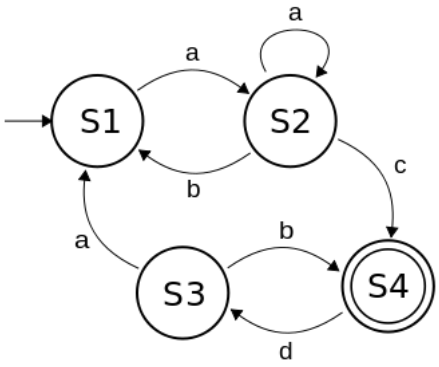


A *hidden Markov model* is the simplest nontrivial dynamic Bayes net. A *factorial HMM* is a dynamic BN whose 2-TBN has the structure of $X_i \rightarrow X_i'$ with a single observed variable Y'. A *coupled HMM* is a DBN whose 2-TBN has chains interacting between adjacent chains and each having its own private observed variable Y'.

5.1.2 State-observation Models

Another way to view temporal processes is a *state-observation model*. We seperate the system dynamics from the observations, which is useful if observations are obtained from a noisy sensor. We define two components: the transition model $P(X'|X)$ and the observation model $P(O|X)$. We will consider two state-observation models: HMM and linear dynamical systems.

Although HMMs are a type of simple DBN, it encodes structure left implicit by a DBN. The transition model $P(S'|S)$ is assumed to be sparse, with may transitions $P(s'|s)$ with zero probability. An alternative way to represent an HMM is shown below. This is analogous to a probabilistic finite-state automaton.



In most cases the observation model is deterministic $P(o|s) = 1$. In other cases, we may have sensors that work with probability p. In this case, $P(o|s) = p$.

Despite being the simplest temporal model, HMMs are extremely useful. It is the key technology in all speech-recognition systems. There are three distinct layers: the language model, the word model, and the acoustic model. The language model represents the distribution over sequences of words. Usually, a bigram model $P(W_i|W_{i-1})$ suffices. The word model describes the composition of individual words in terms of phonemes. The International Phonetic Alphabet contains about 100 phonemes, and each word is composed of these phonetic units. The acoustic model maps short time segments (around 10-25ms) to phonemes. A *hierarchical HMM* of these three layers defines the joint distribution over words, phonemes, and acoustic units. Each state has the form (w, i, j). where w is the current word, i is the phoneme, and j is the sound interval. This model obtains state-of-the-art performance.

A *linear dynamical system*, sometimes called Kalman filters, can be viewed as a dynamic Bayesian network where all variables are continuous and dependencies are linear Gaussians.

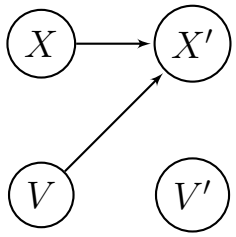


Figure 16: Vehicle position Scenario

Say we want to model a moving vehicle over time. The transition model would have 2 CPDs, $P(X'|X, V) = \mathcal{N}(X + V\Delta; \sigma_X^2)$ and $P(V'|V) = \mathcal{N}(V; \sigma_V^2)$, where Δ is the length of our time slice. The observation model is a noisy measurement of X, $P(O|X) = \mathcal{N}(X; \sigma_X^2)$.

Traditionally, linear dynamical systems are represented by vector-valued random variables.

$$P(X^{(t)}|X^{(t-1)}) = \mathcal{N}(AX^{(t-1)}; Q)$$
$$P(O^{(t)}|X^{(t)}) = \mathcal{N}(HX^{(t)}; R)$$

We can generalize the linear dynamical model by allowing transitions be any function. An *extended Kalman filter* is a system where state and observation variables are real-valued vectors, while state transition and observation models are nonlinear functions

$$P(X^{(t)}|X^{(t-1)}) = f(X^{(t-1)}; U^{(t-1)})$$
$$P(O^{(t)}|X^{(t)}) = g(X^{(t)}; R^{(t-1)})$$

Instead of using stochastic CPDs to model the system, we split the model in terms of deterministic functions and noise.

What if we want to model a vehicle that switches lanes? Depending on whether it stays, turns right, or turns left, the dynamics of the system will change. In a *switching linear dynamical system* (SLDS), we can switch between different modes. We introduce a "mode" variable D that is a parent of some/all variables in the model.

5.2 Template Variables and Template Factors

Dynamic Bayesian networks are only one type of template-based model. We will now provide the fundamental building blocks that generalizes to not only temporal models but all template-based models.

Say we want to model a student's grade in certain courses. Each student and course is an *object*, an instantiation of the Student and Course *class*. The *template variable* or *attribute* can instantiate a specific random variable with a tuple of *arguments*, eg Grade(George, CS170). Hence, template attributes are a "generator" for random variables in a given probability space. A *relation* is a property of a set of objects. Took-course might be a relation between students and courses.

Given a set of template attributes, we can produce an infinite set of probability models depending on different instantiations of objects. For example, we can define attributes Grade(student, course), SAT(student), Diffuiculty(course). The probability space of objects $(S = \{George, Alice\}, C = \{CS70, MATH110\})$ is different from that of objects $(S = \{George, Betty, Carol\}, C = \{CS170, STAT134\})$, but they both use the same template attributes.

An *object skeleton* K specifies a finite fixed set of objects $O^K[Q]$

$$O^K[U_1...U_k] = O^K[Q_1[U_1]] * ... * O^K[Q_k[U_k]]$$

where U_i is a typed logical variable and $Q[U_i]$ is the class associated with the variable. We can just treat U_i as a reference to the class Q_i. An *attribute* is a function $A(U_1, ..., U_k)$ and $U_1...U_k$ is called the *argument signature* denoted $\alpha(A)$.

We define sets of *ground random variables*

$$\mathcal{X}_K[A] = \{A(\gamma) : \gamma \in \Gamma_K[A]\}$$
$$\mathcal{X}_K[\mathbb{N}] = \cup_{A \in \mathbb{N}} \mathcal{X}_K[A]$$

where $\Gamma_K[A]$ is the set of possible assignments to the logical variables in the argument signature of A. The ground random variables are the instantiated random variables of the distribution, eg (George, CS170). Each set of ground

random variables defines a probability space over different object skeletons, an observation observed in the above example.

There are limitations to this model. First, we may want to provide additional information between objects via relations, such as the structure in a family tree. Second, we assumed the number of objects is fixed. In certain domains, we may be uncertain about this number

Lastly, we want to define the CPD or factors over the instantiated ground variables. A *template factor* is a function ξ defined over a tuple of template attributes $A_1...A_l$. It is a mapping between $Val(A_1) * ... * Val(A_l) \mapsto \mathbb{R}$. An *instantiated factor* is a particular instance of a template factor.

5.3 Directed Probabilistic Models for Object Relational Domains

Using the framework of the previous section, we can specify a template-based representation language that can encode directed probabilistic models.

5.3.1 Plate Models

In a *plate model*, an object type is called a plate. There may be several instantiations of a plate within the *ground Bayesian Network*. Say we want to model how intelligence $I(s)$ affects grades $G(s)$. The following is the plate model and ground Bayesian network.

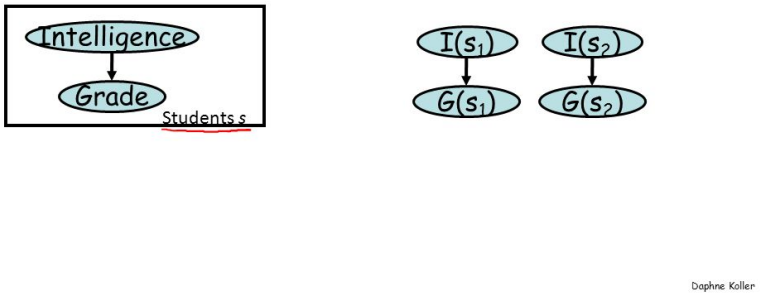


Figure 17: Student example with single plate

A single plate doesn't show the full expressive power of plate models. Say we want to factor in a course's difficulty. The representation would require a nested plate

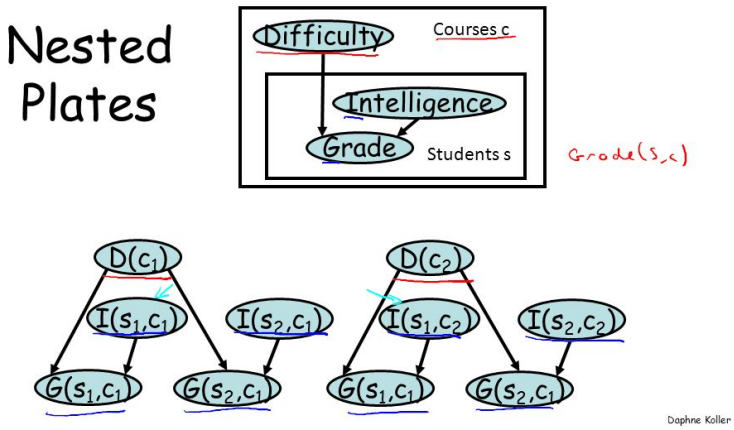


Figure 18: Student example with nested plates

We might want to represent an intelligence attribute to take only a student as an argument, not a (student, course) pair. We can separate them using overlapping plates

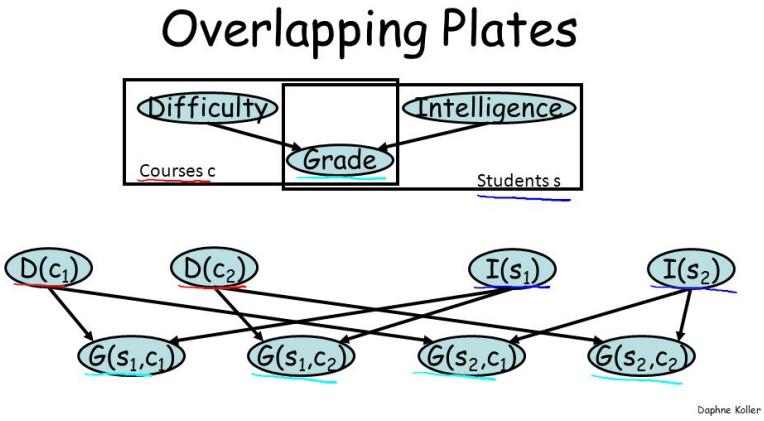


Figure 19: Student example with overlapping plates

Plate models provide a language for encoding models with repeated structure and stored parameters. They can lead to more informed and computationally inexpensive inferences. They can also specify infinite sets, as the set of possible objects is infinite.

A *plate model* defines, for each template attribute $A \in \mathbb{N}$ with argument signature $U_1 ... U_k$

- a set of *template parents*

$$Pa_A = \{B_1(U_1)...B_l(U_l)\}$$

- a template CPD $P(A|Pa_A)$

Attributes can only depend on other attributes of the same parents. We allow plates to intersect on shared attributes. A plate model and object skeleton K defines a *ground Bayesian network* $\beta_K^{M_{Plate}}$. Let $A(U_1...U_k)$ be many template attributes in N. Then, for any assignment $\gamma =< U_1 \mapsto u_1...U_k \mapsto u_k >\in \Gamma[A] >$, we have $A(\gamma)$ in the grounded network, with $B(\gamma)$ for all $B \in Pa_A$ as parents and instantiated CPD $P(A(\gamma)|Pa_A(\gamma))$.

The plate model is limited by the constraint: attributes can only depend on attributes of the same object. This prevents us from being able to create a geneology tree because the $Genotype(child)$ is dependent on the $Genotype(mother)$. Similarly, we can't specify temporal models because the car's current position is dependent on a car's previous position/velocity.

5.3.2 Probabilistic Relational Models

In the Genotype example, we might want to allow Genotype(U) to depend on $Genotype(U')$. However, we must specify for which objects the dependency is applicable. We can specify a guard like $Mother(U, U')$ that the dependency is contingent on.

For a template attribute A, we define a *contingency dependency model* as a tuple consisting of:

- A parent argument signature $\alpha(Pa_A)$ which is a tuple of typed logical variables U_i such that $\alpha(Pa_A) \supseteq \alpha(A)$
- A guard Γ which is a binary-valued formula defined in terms of a set of template attributes Pa_A^Γ over the argument signature $\alpha(Pa_A)$
- a set of template parents $Pa_A = \{B_1(U_1)...B_l(U_l)\}$

The contingency dependency model lets each attribute be applicable to only certain objects through the use of "guards".

A *probabilistic relational model* (PRM) $\mathcal{M}_{PRM}$ defines a contingency dependency model and template CPD for each attribute $A \in N$. The difference between the plate model and PRM is the logical attributes of the parents do not necessarily need to be a subset of that of the child.

A PRM M_{PRM} and object skeleton K define a ground Bayesian network $\beta_K^{M_{PRM}}$ as follows. For any assignment $\gamma \in \Gamma_K[A]$, we have a variable $A(\gamma)$ in the ground network. This varaible has, for any $B \in Pa_A^\Gamma \cup Pa_A$ and any assignment γ' to $\alpha(Pa_A) - \alpha(A)$, the parent that is $B(\gamma, \gamma')$.

One issue with allowing attributes of one object depend on that of another is that dependency structure may be arbitrarily large. We can solve this issue by defining a *relational skeleton* K_r that defines a certain set of relationships between objects. Since guards and therefore CPDs will be bounded by these relations, we have successfully restricted the # of CPDs in the PRM. The relational skeleton also addresses the problem of cycles. Since the relational structure is usually relatively simple, the resulting grounded BN is acyclic.

When defining the CPD, we must allow for variable number of parents. Not only that, we must also have a symmetric CPD because each instantiation of objects are interchangeable. The two approaches are a symmetric CPD or *aggregator CPD*. For example, we may use a noisy-or CPD or $min(parentValues)/max(parentValues)$. This defines the relation between the parents $B \in Pa_A$ and child A.

We may want a test at a template-levegl whether the generated ground BN has a possibility of being cyclic. Fortunately, if there is no cycle in the *template dependency graph*, there will be no cycle in the grounded network. A *template dependency graph* M_{PRM} contains a node for each template-level attribute A, and directed edge $B \rightarrow A$ if $B \in Pa_A^\Gamma \cup Pa_A$.

5.4 Undirected Representation

The undirected equivalent is fairly similar to direted template-based models. A relational Markov network M_{RMN} is defined in terms of a set Λ of *template features* where each $\lambda in \Lambda$ comprises:

- a real-valued *template feature* f_λ whose arguments are $N(\lambda) = \{\Lambda_1(U_1)...\Lambda_l(U_l)\}$
- a weight on the feature $w_\lambda \in \mathbb{R}$

Given relational Markov Network M_{RMN} and an object skeleton K, we can define a *grounded Gibbs distribution* $P_K^{M_{RMN}}$ as follows:

- variables in the network are $\mathcal{X}_K[N]$
- $P_K^{M_{RMN}}$ contains a term $exp(w_\lambda * f_\lambda(\gamma))$ for each feature template $\lambda \in \Lambda$ and each assignment $\gamma \in \Gamma_K[\alpha(\lambda)]$

The Gibbs distribution defines a grounded Markov Network, where we connect every pair of variables that appear together in some factor.

One issue is the right aggregated function of the log-linear model. Each occurrence of a feature has a log-linear contribution to the unnormalized probability density. Unfortunately, this linear aggregation behavior may not be applicable in certain domains. Consider a "viral marketing" example - where a social network of individuals related by $Friends(P, P')$ has interest in a gadget modeled by $Gadget(P)$. In a log-linear model, the unnormalized probability that a person is interested in the gadget grows log-linearly as with the number of friends interested in the gadget. A more realistic model will account for the "saturation effect" where the impact of friends' interests diminish as the number increases.

Another issue is the dense connectivity. We can fix this issue by defining a relational skeleton and simplifying the model.

A final issue only arising in undirected models is the global influence factors can have. Since introducting a new object can drastically change the factors/values of the current model, it is problematic to use one instance of a template-based model to help construct another instance.

Application: collective classification of webpages
In a university sebsite, we may want to classify pages by "professor", "student", "project" using words within the webpage. However, we can refine our classification by incorporating links into the model: "professor-proj", "student-proj" links can be set to have higher clique potentials. Incorporating links expands the classification task to not a single webpage but the entire collection of webpages.

- Input: object with features, entity relations
- Output: collective classification of objects

5.5 Structural Uncertainty

One of the biggest issues facing template-based models is *structural uncertainty*. Distributions in the real world may not have set number of objects and relations. Different models will handle structural uncertainty differently. First, we will define two types of structural uncertainty: *relational uncertainty* and *object uncertainty*.

5.5.1 Relational Uncertainty

Say that we are creating a relational skeleton that is bipartite for which professor teaches a course. There are 10 professors and 20 courses. However, the relation between professors and courses is stochastic. How can we model uncertainty in relations?

One approach is to model each relation independently. The relation $Teaches(P, C)$ is defined to be true with a probability 0.1. The expected number of courses per professor is 2, and expected number of professors for a course is 1. So far, everything seem reasonable. With further analysis, however, we can see this model is not realistic. The probability that a particular professor teaches l courses follows a binomial distribution $\binom{20}{l}0.1^l 0.9^{20-l}$, so the probability that at least one professor teaches 5 or more courses is 29%. The issue with this approach is that each relation is chosen independently, when in reality $Teaches(P, C)$ affects the distribtuion of another $Teaches(P, C')$.

Another approach is to introduce set-valued functions, a function that maps one set of objects to another. In the above example, we might define a function $CoursesOf(p)$ that maps to all courses taught by p. Though this allows dependencies between relations, it still allows the possibility that a single courses may be taught by 10 professors.

There is no natural way to guarantee structural properties of a relation.

Luckily, this discussion only applies to directed template structures. An undirected framework can define a template feature ascribing a high probability that a set of relations is true. A low potential for template event

$$Teaches(P, C) = true, Teaches(P', C) = true$$

will enforce the constraint that at most one professor teaches a course.

5.5.2 Object Uncertainty

Sometimes we won't know the set of objects the world may have. For example, a single person may be a student 34 in CS101, student 57 in Econ203, eldest daughter of John and Mary, and so on. Should we treat this person as 3 objects or 1?

A very natural solution is to introduce *reference objects* r. We can define a relation $RefersTo(r, o)$ that is true whenever r refers to an object o. Alternatively, we can efine an object-valued function $Reference(r) \mapsto o$.

Attribute similarity potentials measure how similar a reference object is to the true object. For example, "John Franklin Adams" might go by "JF Adams" in one context and "Frank Adams" in another. Trying to determine which reference corresponds to an object is known as the *correspondence problem.*

An alternative approach is to eliminate the "true object" and have references point to each other. We can introduce the binary relation $SameAs(r, r')$ To ensure consistency, $SameAs$ satisfies the following properties:

- Reflexivity: $SameAs(r, r)$
- Symmetry: $SameAs(r, r')$ iff $SameAs(r', r)$
- Transivity: $SameAs(r, r')$ and $SameAs(r', r'')$, then $SameAs(r, r'')$

This approach has several problems. There is no natural way to put factors that should apply once per entity. For example, say we want to define a template factor on $\{Name(R), Gender(R)\}$. Then, the factor will apply to multiply times to the same person if they have multiple references. The same sort of "multiplicative problem" will occur during inference.

5.6 Conclusion

- Template-based representations allow us to encode potentially infinite set of distributions
- They are useful for inter-object reasoning
- Inference at a template-level is much more efficient than at the ground network level
- Relational language such as function symbols, quantifiers, etc. can give us more expressive firepower

6 Gaussian Network Models

Our discussion so far has been about discrete random variables. Gaussians are a simple way to encode continuous random variables with two glaringly simple assumptions: exponential decay of probability density away from the mean and linearity between variables. These assumptions allow Gaussians to be modeled easily. Surprisingly, Gaussians provide a good estimate for many real world applications. It can also be extended to other distribtuions like Guassian mixtures or nonlinear interactions.

6.1 Multivariate Gaussians

The density function of a *multivariate Gaussian* is definedby its mean vector μ and covariance matrix Σ.

$$P(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{0.5}} exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$

To guarantee a pdf that integrates to 1, Σ must be *positive definite*

$$\forall x \in \mathbb{R}^n, x^T\Sigma x > 0$$

This also guarantees a nonzero determinant $|\Sigma|$ since positive definite matrices are nonsingular.

The inverse covariance matrix $I = \Sigma^{=1}$ is sometimes called the *information matrix* and we can use it to get the alternative form of the Gaussian density function:

$$\begin{aligned} -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) &= -\frac{1}{2}(x-\mu)^T I(x-\mu) \\ &= -\frac{1}{2}[x^TJx - 2x^TJ\mu + \mu^TJ\mu] \end{aligned}$$

The last term is a constant so

$$p(x) \propto exp[-\frac{1}{2}x^TJx + (J\mu)^Tx]$$

which is called *information form.* $h = J\mu$ is called the *potential vector.*

There are two operations we want to perform with the Gaussian pdfs:

- Compute the marginal AY from joint distribution $\{X, Y\}$. Since

  $$p(X,Y) = \mathcal{N}(\begin{bmatrix}\mu_X \\ \mu_Y\end{bmatrix}; \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY}\end{bmatrix})$$

  it is evident that $Y \sim \mathcal{N}(\mu_Y; \Sigma_{YY})$

- Conditioning on $Z = z$ where $Z \subseteq X$. We can jthe evidence into the pdf and get a new pdf. It will be a Gaussian with parameters of the subset of $X - Z$.

Independencies are explicitly encoded within the Gaussian pdf X_i and X_j are independent iff $\Sigma i, j = 0$.

6.2 Gaussian Bayesian Network

Now, we have explicitly described the Gaussian probability distribution, we can discuss how to encode this distribution to a directed model. A *Gaussian Bayesian network* is a BN whose variables are continuous and CPDs are linear Gaussians. A Gaussian BN will always define a joing Gaussian distribution. Let Y be a linear Gaussian of parents $X_1...X_n$.

$$P(Y|x) = \mathcal{N}(\beta_0 + \beta^T x; \sigma^2)$$

If $X_1...X_k$ are jointly Gaussian $\mathcal{N}(\mu; \Sigma)$, then

- $Y \sim \mathcal{N}(\mu_Y; \sigma_Y^2)$, where

$$\mu_Y = \beta_0 + \beta^T \mu$$
$$\sigma_Y^2 = \sigma^2 + \beta^T \Sigma \beta$$

- The joint distribution $\{X, Y\}$ is normal where

$$Cov[X_i; Y] = \sum_{j=1}^{k} \beta_j \Sigma_{i,j}$$

Through induction, we can see that a Gaussian BN will be a Gaussian.

The converse is also true; conditioning a Gaussian results in a normal distribution where there is a linear dependence on conditioned variables. Let $\{X, Y\}$ have a joint Gaussian distribution. Then the conditional density

$$P(Y|X) = \mathcal{N}(\beta_0 + \beta^T X; \sigma^2)$$

where

$$\beta_0 = \mu_Y - \Sigma_{YX}\Sigma_{XX}^{-1}\mu_X \beta = \Sigma_{XX}^{-1}\Sigma_{YX}\sigma^2 = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}$$

These theorems allow us to efficiently transform a Gaussian to a linear Gaussian and vice versa. Both are relatively compact and is useful for different tasks.

6.3 Gaussian Markov Field

A Gaussian distribution can also be represented as a Markov Random Field (MRF). First, we use the information form to break up the terms:

$$P(x) \propto exp[-\frac{1}{2}x^T Jx + (J\mu)^T x]$$

into single-variable part

$$-\frac{1}{2}J_{i,i}x_i^2 + h_i x_i$$

and pairs of variables

$$-\frac{1}{2}[J_{i,j}x_i x_j + J_{j,i}x_j x_i] = -J_{i,j}x_i x_j$$

This induces a pairwise Markov Network where node potentials are derived from the single-variable part, dependent on the diagonal elements of the information matrix, and edge potentials are derived from the pairwise part, dependent on off-diagonal elements of the information matrix. This network is called a *Gaussian MRF* (GMRF).

To convert a pairwise Markov Network to a multivariate Gaussian distribution, we can write node and edge potentials as follows, where d_i and a_i are learned/solved coefficients.

$$\epsilon_i(x_i) = d_1^i + d_1^i x_i + d_2^i x_i^2 \epsilon_{i,j}(x_i, x_j) = a_{0,0}^{i,j} + a_{0,1}^{i,j} x_i + a_{1,0}^{i,j} x_j + a_{1,1}^{i,j} x_i x_j + a_{2,0}^{i,j} x_i^2 + a_{0,2}^{i,j} x_j^2$$

By aggregating terms, we can reformulate any such set of potentials in the log-quadratic form

$$P(X) = exp[-\frac{1}{2} x^T J x + h^T x]$$

where J is symmetric. This Markov network is a Gaussian density iff J is positive definite.

Unfortunately, there is no way to test whether a GMRF encodes a valid Gaussian, besides computing J and checking whether it's positive definite. In particular, there are no local tests that can be applied to network parameters that precisely characterizes a valid Gaussian. There are, however, sufficient tests that can induce a valid density.

One simple test is performed by checking the information matrix. A *quadratic MRF* is said to be *diagonally dominant* iff for all i

$$\sum_{j \neq i} |J_{i,j}| < J_{i,i}$$

If an MRF is diagonally dominant, it defines a valid Gaussian MRF.

Another test is to check whether each pairwise edge potential is normalizeable. A quadratic MRF is said to be *pairwise normalizeable* if

- for all i, d_2^i ¿ 0
- for all i, j, the 2x2 matrix

$$\begin{bmatrix} a_{02}^{i,j} & a_{11}^{i,j}/2 \\ a_{11}^{i,j}/2 & a_{20}^{i,j} \end{bmatrix}$$

  is positive semidefinite

If a pairwise MRF is pairwise normalizeable, it is a GMRF. These tests only test a particular parameterization of the MRF. It is completely possible that a different parameterization of the same density induces a valid GMRF.

6.4 Conclusion

- The three representational classes, multivariate Gaussians, linear Gaussian BN and GMRFs, are equivalent.
- The undirected models have a particularly elegant connection to Gaussians, as zeros in the information matrix correspond precisely to missing edges in the minimal I-map Markov network.

7 Exponential Family

In the past chapters, we focused on representing a single distribution. Now, we will consider a family of distributions to lay a theoretical groundwork for learning and inference. This chapter is somewhat abstract and heavily mathematical.

7.1 Exponential Families

An *exponential family* $\mathcal{P}$ over $\mathcal{X}$ is specified by four components.

1. a *sufficient statistic function* τ from assignments to $\mathcal{X}$ to $\mathcal{R}^K$
2. a *parameter space* that is a convex set $\Theta \subseteq \mathcal{R}^M$ of legal parameters
3. a *natural parameter function* t mapping parameters in $\mathcal{R}^M$ to distribution in $\mathcal{R}^K$.
4. an *auxiliary measure* A over $\mathcal{X}$

Each vector of parameters $\theta \in \Theta$ specifies a distribution P_θ in the family at

$$P_\theta(\xi) = \frac{1}{Z(\theta)} A(\xi) exp[< t(\theta), \tau(\xi) >]$$

where $< t(\theta), \tau(\xi) >$ is the inner product of the vectors $t(\theta)$ and $\tau(\xi)$ and

$$Z(\theta) = \sum_{\xi} A(\xi) exp[< t(\theta), \tau(\xi) >]$$

is the partition function of P, which must be finite. The parametric family $\mathcal{P}$ is defined as

$$\mathcal{P} = \{P_\theta : \theta \in \Theta\}$$

Intuitively, a parameter vector θ determines a member of the family. The sufficient statistic function τ summarizes the aspect of an instance that are relevant for assigning probability. The function t maps parameters to the space of sufficient statistics. An auxiliary measure A assigns additional preferences among instances not dependent on parameters.

The Bernoulli distribution is in the exponential family. We can set

$$\tau(X) =< 1\{X = x^1\}, 1\{X = x^0\} >$$

a numerical vector representation of the value of X, and

$$t(\theta) =< ln\theta, ln(1-\theta) >$$

It's easy to see that for when $X = x^1$, $\tau(X) =< 1, 0 >$ and so

$$exp\{< t(\theta), \tau(X) >\} = e^{1*ln\theta + 0*ln(1-\theta)} = \theta$$

Similarly, when $X = x^0$, $exp\{< t(\theta), \tau(X) >\} = 1 - \theta$. So when $Z(\theta) = 1$, this representation is the Bernoulli distribution.

Most parameterized distribution encountered in probability books are in teh exponential family, including Poisson distributions, exponential distributions, geometric distributions, and gamma distributions.

We enforce additional constraints to the exponential families to help with inference and learning.

1. Parameter space Θ must be "well-behaved", aka convex and open in $\mathcal{R}^M$
2. Each parameter family must be nonredundant, or

$$\theta \neq \theta' \Rightarrow P_\theta \neq P_{\theta'}$$

We can check for nonredundancy by checking if function t is invertible.

7.1.1 Linear Exponential Families

A *linear exponential family* is an exponential family where the natural parameter function t is the identity function. This means the parameters are *natural parameters*, parameters with the same dimension K as the representation of the data. A linear exponential family has the form

$$P_\theta(\xi) = \frac{1}{Z(\theta)} exp[< \theta, \tau(\xi) >]$$

The *natural parameter space* is the space of allowed parameters for a certain sufficient statistic function τ where the pdf is legal (normalizeable) or

$$\Theta = \{\theta \in \mathcal{R}^K : \int exp[< \theta, \tau(\xi) >] d\xi < \infty\}$$

A linear exponential family must have an open, convex natural parameter space.

All exponential families can be turned into linear exponential families by representing θ by $t(\theta)$. This is helpful because linear exponential families are simpler; we only need to define τ because all other variables are implicitly defined. However, this isn't always trivial.

Consider the Bernoulli distribution where $t(\theta) =< ln\theta, ln(1-\theta) >$. This curve is not an open, convex set, so setting $\theta' = t(\theta)$ would not be a natural parameter space.

Alternatively, we might want to use a parameter space in two dimensions, corresponding to the sufficient statistic function $\tau(X) =< 1\{X = x^1\}, 1\{X = x^0\} >$, which gives

$$\begin{aligned} P_\theta(x) &\propto exp[< \theta, \tau(x) >] \\ &= exp[\theta_1 1\{X = x^1\} + \theta_2 1\{X = x^2\}] \end{aligned}$$

Unfortunately, this family is redundant. For every constant c, the parameters $[\theta_1 + c, \theta_2 + c]$ defines the same distribution as $[\theta_1, \theta_2]$.

Thus the 1d case is not well-behaved, while the 2d case overparameterizes. Fortunately, we can use an alternate 1d representation

$$\begin{aligned} \tau(x) &= 1\{X = x^1\} \\ t(\theta) &= ln\frac{\theta}{1-\theta} \end{aligned}$$

and so

$$Z(\theta) = 1 + \frac{\theta}{1-\theta} = \theta$$

Thus,

$$P_\theta(x^1) = (1-\theta)\frac{\theta}{1-\theta} = \theta$$

which is a linear exponential representation of the Bernoulli distribution.

Are there families that are not linear? Yes, but we will discuss that in another chapter.

7.2 Factored Exponential Families

So far, we have only discussed univariate probability distributions. What about distributions with multiple random variables? Consider the log-linear model

$$P(X_1...X_n) \propto exp[\sum_{i=1}^{k} \theta_i * f_i(D_i)]$$

This distribution is a linear exponential family where

$$\tau(\xi) = [f_1(d_1)...f_k(d_k)]$$

Since all Markov networks can be represented as a log-linear model, this suffices to show the family of MNs are linear exponential.

Exponential families with these product forms are called *exponential factor families* where a *factor* is

$$\phi_\theta(\xi) = A(\xi)exp[< t(\theta), \tau(\xi) >]$$

and the *composition of factors* $\phi_1...\phi_k$ parameterized by $\theta_1...\theta_k$ is

$$\begin{aligned} P_\theta(\xi) &\propto \prod_i \phi_{\theta_i}(\xi) \\ &= (\prod_i A_i(\xi))exp[\sum_i < t_i(\theta_i), \tau_i(\xi) >] \end{aligned}$$

We can see that the composition of exponential factors is an exponential family with $\tau(\xi) = \tau_1(\xi) \circ ... \circ \tau_k(\xi)$ and natural parameters $t(\theta) = t_1(\theta) \circ ... \circ t_k(\theta_k)$. This can be easily extended to linear exponentials: the composition of linear exponential factors is a linear exponential family.

Bayesian networks also follows the exponential form through the product of CPDs. However, it turns out a BN does not define a linear exponential family. In general, any Bayesian network that contains immoralities does not induce a linear exponential family.

7.3 Entropy and Relative Entropy

Comparing entropy and maximizing entropy subject to constraints is useful in many tasks. Characterizing entropy within exponential families will allow us to efficiently perform both tasks.

Let P_θ be a distribution in an exponential family. Then

$$H_{P_\theta}(\mathcal{X}) = lnZ(\theta) - < E_{P_\theta}[\tau(\mathcal{X})], t(\theta) >$$

We can see that entropy is only dependent on the expectation of sufficient statistics $\tau(\mathcal{X})$. The entropy of a Markov network can be found in this form. If $P(\mathcal{X}) = \frac{1}{Z}\prod_k \phi_k(D_k)$ is a Markov network.

$$H_P(\mathcal{X}) = lnZ + \prod_k E_P[-ln\phi_k(D_k)]$$

Since the number of joint assignments are exponential, we use clique potentials to specify entropy. Note that this form is dependent on the global computation of Z and marginals for each clique $P(D_k)$ which are nontrivial.

The entropy of a BN can also be found in a similar fashion. If $P(\mathcal{X}) = \prod_i P(X_i|Pa_i^{\mathcal{G}})$ is a distribution consistent with Bayesian network $\mathcal{G}$. then

$$\begin{aligned} H_P(\mathcal{X}) &= E_P[-lnP(\mathcal{X})] \\ &= E_P[-\sum_i ln(P(X_i|Pa_i^{\mathcal{G}})] \\ &= \sum_i E_P[-lnP(X_i|Pa_i^{\mathcal{G}})] \\ &= \sum_i H_P(X_i|Pa_i^{\mathcal{G}}) \end{aligned}$$

Each conditional entropy $H_P(X_i|Pa_i^{\mathcal{G}})$ can be found

$$H_P(X_i|Pa_i^{\mathcal{G}}) = \sum_{pa_i^{\mathcal{G}}} P(pa_i^{\mathcal{G}})H_P(X_i|pa_i^{\mathcal{G}})$$

by a weighted average over entropies $H_P(X_i|pa_i^{\mathcal{G}})$. Each weighting term $P(pa_i^{\mathcal{G}})$. is dependent on the entire joint distribution and other CPDs, so we cannot simply multiply local entropy calculations to find the total joint entropy.