CS 188 Introduction to Artificial Intelligence
Spring 2021 Note 4

These lecture notes are heavily based on the textbook Artificial Intelligence: A Modern Approach and the notes originally written by Josh Hug and Jacky Liang.

Introduction to Probability

We're assuming that you've learned the foundations of probability in CS70, so these notes will assume a basic understanding of standard concepts in probability like PDFs, conditional probabilities, independence, and conditional independence. Here we provide a brief summary of probability rules we will be using.

A probability model assigns values $P(\omega)$ to each world $\omega \in \Omega$ where Ω is the set of all possible worlds. Probability models must satisfy the following conditions

$$0 \leq P(\omega) \leq 1$$
$$\sum_{\omega} P(\omega) = 1$$

$P(A)$ denotes the **probability distribution** of the variable A. For instance if A is a binary variable then $P(A=0) = p$ and $P(A=1) = 1-p$ for some $p \in [0,1]$. We use the notation $P(A,B,C)$ to denote the joint distribution of the variables A,B,C. In joint distributions ordering does not matter i.e. $P(A,B,C) = P(C,B,A)$. The **marginal** distribution of A,B can be obtained by summing out all possible values that variable C can take as $P(A,B) = \sum_C P(A,B,C)$. The marginal distribution of A can also be obtained as $P(A) = \sum_B \sum_C P(A,B,C)$. **Conditional** probabilities assign probabilities to events conditioned on some known facts. For instance $P(A|B=b)$ gives the probability distribution of A given that we know the value of B equals b. Conditional probabilities are computed using **Bayes' rule** also known as **Bayes' theorem**. Bayes' rule gives us the conditional probability of A given B using the joint distribution of A,B and the marginal distribution of B as

$$P(A|B) = \frac{P(A,B)}{P(B)}.$$

If we know the joint distribution of A,B,C and we want to condition on B,C then from Bayes' rule we obtain

$$P(A|B,C) = \frac{P(A,B,C)}{P(B,C)}.$$

Finally if we have a conditional distribution $P(A,B|C)$ and we want to further condition on the event B then from Bayes' rule we obtain

$$P(A|B,C) = \frac{P(A,B|C)}{P(B|C)},$$

i.e. the event we had already conditioned on (C) appears in the conditional part of all probabilities in Bayes' rule. The **chain** rule repeatedly applies the product rule (which is just a reordering of the terms of Bayes' rule) to obtain $P(A,B,C) = P(A,B|C)P(C) = P(A|B,C)P(B|C)P(C)$.

Regarding independence, we say that A and B are independent when $P(A,B) = P(A)P(B)$. Furthermore, if A and B are conditionally independent given C then $P(A,B|C) = P(A|C)P(B|C)$. Finally, the phrase A is independent of B given C means that $P(A|B,C) = P(A|C)$.

Probabilistic Inference

In artificial intelligence, we often want to model the relationships between various nondeterministic events. If the weather predicts a 40% chance of rain, should I carry my umbrella? How many scoops of ice cream should I get if the more scoops I get, the more likely I am to drop it all? If there was an accident 15 minutes ago on the freeway on my route to Oracle Arena to watch the Warriors' game, should I leave now or in 30 minutes? All of these questions (and innumerable more) can be answered with **probabilistic inference**.

In previous sections of this class, we modeled the world as existing in a specific state that is always known. For the next several weeks, we will instead use a new model where each possible state for the world has its own probability. For example, we might build a weather model, where the state consists of the season, temperature and weather. Our model might say that $P(winter, 35^\circ, cloudy) = 0.023$. This number represents the probability of the specific outcome that it is winter, 35°, and cloudy.

More precisely, our model is a **joint distribution**, i.e. a table of probabilities which captures the likelihood of each possible **outcome**, also known as an **assignment**. As an example, consider the table below:

Season	Temperature	Weather	Probability
summer	hot	sun	0.30
summer	hot	rain	0.05
summer	cold	sun	0.10
summer	cold	rain	0.05
winter	hot	sun	0.10
winter	hot	rain	0.05
winter	cold	sun	0.15
winter	cold	rain	0.20

This model allows us to answer questions that might be of interest to us, for example:

- What is the probability that it is sunny? $P(W = sun)$
- What is the probability distribution for the weather, given that we know it is winter? $P(W \mid S = winter)$
- What is the probability that it is winter, given that we know it is rainy and cold? $P(S = winter \mid T = cold, W = rain)$
- What is the probability distribution for the weather and season give that we know that it is cold? $P(S,W \mid T = cold)$

Given a joint PDF, we can trivially perform compute any desired probablity distribution $P(Q_1 \dots Q_k \mid e_1 \dots e_k)$ using a simple and intuitive procedure known as **inference by enumeration**, for which we define three types of variables we will be dealing with:

1. **Query variables** Q_i, which are unknown and appear on the left side of the probability distribution we are trying to compute.
2. **Evidence variables** e_i, which are observed variables whose values are known and appear on the right side of the probability distribution we are trying to compute.
3. **Hidden variables**, which are values present in the overall joint distribution but not in the distribution we are currently trying to compute.

In this procedure, we collect all the rows consistent with the observed evidence variables, sum out all the hidden variables, and finally normalize the table so that it is a probability distribution (i.e. values sum to 1).

For example, if we wanted to compute $P(W \mid S = winter)$, we'd select the four rows where S is winter, then sum out over T and normalize. This yields the following probability table:

W	S	Unnormalized Sum	Probability
sun	winter	$0.10 + 0.15 = 0.25$	$0.25/(0.25 + 0.25) = 0.5$
rain	winter	$0.05 + 0.20 = 0.25$	$0.25/(0.25 + 0.25) = 0.5$

Hence $P(W = sun \mid S = winter) = 0.5$ and $P(W = rain \mid S = winter) = 0.5$, and we learn that in winter there's a 50% chance of sun and a 50% chance of rain (classic California weather).

So long as we have the joint PDF table, inference by enumeration (IBE) can be used to compute any desired probablity distribution, even for multiple query variables $Q_1 ... Q_k$.

Bayes Nets (Representation)

While inference by enumeration can compute probabilities for any query we might desire, representing an entire joint distribution in the memory of a computer is impractical for real problems - if each of n variables we wish to represent can take on d possible values (it has a **domain** of size d), then our joint distribution table will have d^n entries, exponential in the number of variables and quite impractical to store!

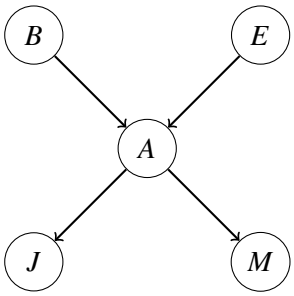
Bayes nets avoid this issue by taking advantage of the idea of conditional probability. Rather than storing information in a giant table, probabilities are instead distributed across a large number of smaller local probability tables along with a **directed acyclic graph** (DAG) which captures the relationships between variables. The local probability tables and the DAG together encode enough information to compute any probability distribution that we could have otherwise computed given the entire joint distribution.

Specifically, each node in the graph represents a single conditional probability table that is stored, and is conditioned on its parents. For example, a node C with parent nodes A and B indicates that we store the probability table $P(C|A,B)$). This is a result of the fundamental simplifying assumption which belies Bayes Net structure: **each node is conditionally independent of all its ancestor nodes in the graph, given all of its parents**. Thus, if we have a node representing variable X, we store $P(X|A_1,A_2,...,A_N)$, where $A_1,...,A_N$ are the parents of X.

As an example of a Bayes Net, consider a model where we have five binary random variables described below:

- B: Burglary occurs.
- A: Alarm goes off.
- E: Earthquake occurs.
- J: John calls.
- M: Mary calls.

Assume the alarm can go off if either a burglary or an earthquake occurs, and that Mary and John will call if they hear the alarm. We can represent these dependencies with the graph shown below.



As a reality check, it's important to internalize that Bayes Nets are only a type of model. Models attempt to capture the way the world works, but because they are always a simplification they are always wrong. However, with good modeling choices they can still be good enough approximations that they are useful for solving real problems in the real world. In general, they will not account for every variable or even every interaction between variables. By making these modeling assumptions, we can exploit graphical structure to produce incredibly efficient inference techniques that are often more practically useful than simple procedures like inference by enumeration.

Returning to our discussion, we formally define a Bayes Net as consisting of:

- A directed acyclic graph of nodes, one per variable X.
- A conditional distribution for each node $P(X|A_1 \dots A_n)$, where A_i is the i^{th} parent of X, stored as a **conditional probability table** or CPT. Each CPT has $n+2$ columns: one for the values of each of the n parent variables $A_1 \dots A_n$, one for the values of X, and one for the conditional probability of X.

In the alarm model above, we would store probability tables $P(B), P(E), P(A \mid B,E), P(J \mid A)$ and $P(M \mid A)$.

Given all of the CPTs for a graph, we can calculate the probability of a given assignment using the chain rule: $P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P(X_i | parents(X_i))$.

For the alarm model above, we might calculate the probability of one event as follows: $P(-b,-e,+a,+j,-m) = P(-b) \cdot P(-e) \cdot P(+a|-b,-e) \cdot P(+j|+a) \cdot P(-m|+a)$.

This works because of the conditional independence relationships given by the graph. Specifically, we rely on the fact that $P(x_i|x_1, \dots, x_{i-1}) = P(x_i|parents(X_i))$. Or in other words, that the probability of a specific value of X_i depends only on the values assigned to X_i's parents.

In later parts we will also be using the term **Markov blanket**. The Markov blanket of a node consists of its parents, children and children's parents. Given its Markov blanket a variable is independent of every other variable in the network.

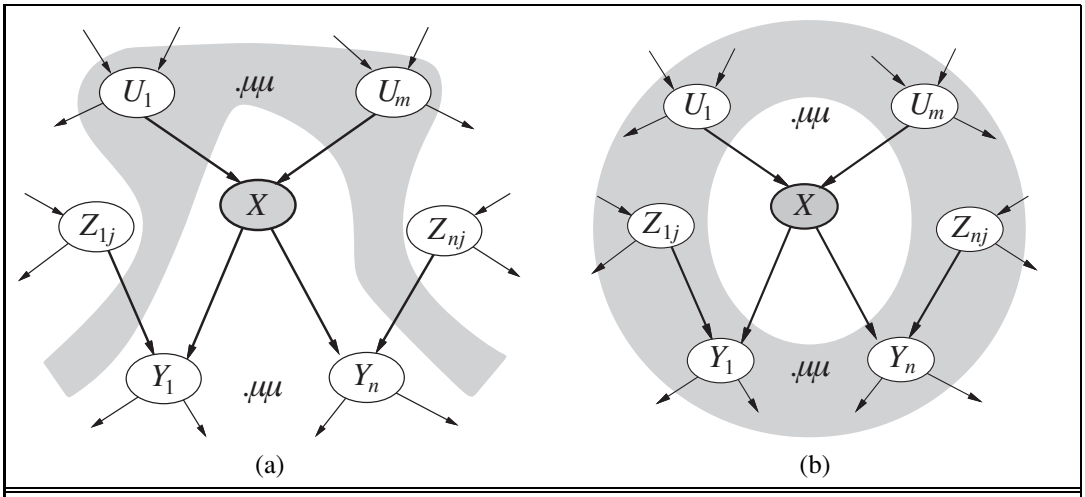


Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

Bayes Nets (Inference)

Inference is the process of calculating the joint PDF for some set of query variables based on some set of observed variables. We can solve this problem naively by forming the joint PDF and using inference by enumeration as described above. This requires the creation of and iteration over an exponentially large table.

An alternate approach is to **eliminate** variables one by one. To eliminate a variable X, we:

1. Join (multiply together) all factors involving X.
2. Sum out X.

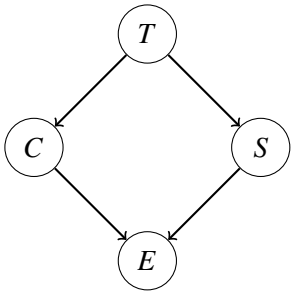
A **factor** is defined simply as an *unnormalized probability*. At all points during variable elimination, each factor will be proportional to the probability it corresponds to but the underlying distribution for each factor won't necessarily sum to 1 as a probability distribution should. The pseudocode can be seen here:

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $factors \leftarrow [\,]$
  for each var **in** ORDER(bn.VARS) **do**
    $factors \leftarrow [\text{MAKE-FACTOR}(var, \mathbf{e}) | factors]$
    if var is a hidden variable **then** $factors \leftarrow \text{SUM-OUT}(var, factors)$
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

Figure 14.11 The variable elimination algorithm for inference in Bayesian networks.

Let's make these ideas more concrete with an example. Suppose we have a model as shown below, where T, C, S, and E can take on binary values, as shown below. Here, T represents the chance that an adventurer takes a treasure, C represents the chance that a cage falls on the adventurer given that he takes the treasure, S represents the chance that snakes are released if an adventurer takes the treasure, and E represents the chance that the adventurer escapes given information about the status of the cage and snakes.



In this case, we have the factors $P(T)$, $P(C|T)$, $P(S|T)$, and $P(E|C,S)$. Suppose we want to calculate the query $P(T|+e)$. The inference by enumeration approach would be to form the 16 row joint PDF $P(T,C,S,E)$, select only the rows corresponding to +e, then summing out C and S and finally normalizing.

The alternate approach is to eliminate C, then S, one variable at a time. We'd proceed as follows:

- Join (multiply) all the factors involving C, forming $P(C,+e|T,S) = P(C|T) \cdot P(+e|C,S)$.
- Sum out C from this new factor, leaving us with a new factor $P(+e|T,S)$.
- Join all factors involving S, forming $P(+e,S|T) = P(S|T) \cdot P(+e|T,S)$.
- Sum out S, yielding $P(+e|T)$.

Once we have $P(+e|T)$, we can easily compute $P(T|+e)$.

While this process is more involved from a conceptual point of view, the maximum size of any factor generated is only 8 rows instead of 16 as it would be if we formed the entire joint PDF.

An alternate way of looking at the problem is to observe that the calculation of $P(+e,T)$ can be done, as it is in inference by enumeration, as follows:

$$\sum_s \sum_c P(T)P(s|T)P(c|T)P(+e|c,s)$$

Variable elimination is equivalent to calculating $P(+e,T)$ as follows:

$$P(T)\sum_s P(s|T)\sum_c P(c|T)P(+e|c,s)$$

Although the previous two expressions compute the same quantity, the second one requires fewer calculations. As a final note on variable elimination, it's important to observe that it only improves on inference by enumeration if we are able to limit the size of the largest factor to a reasonable value. Finally, we call a variable **irrelevant** if it's value has no effect on the query, i.e. if we remove the variable alltogether the value of the query remains unchanged. In general we can remove any leaf node that is not a query or an evidence variable.

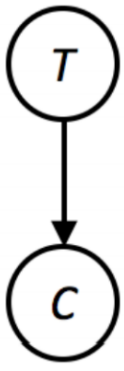
Bayes Nets (Sampling)

An alternate approach for probabilistic reasoning is to implicitly calculate the probabilities for our query by simply counting samples. For example, suppose we wanted to calculate $P(T|+e)$. If we had a magic machine that could generate samples from our distribution, we could collect all samples for which the adventurer escapes the maze, and then compute the fraction of those escapes for which the adventurer also took the treasure. Put differently, if we could run simulations of say, a few million adventurers, we'd easily be able to compute any inference we'd want just by looking at the samples. We call a probability estimate using sampling methods **consistent** if in the large sample limit it becomes exact, i.e. if we run the sampling algorithm many times we get the estimated probability approaching the true probability.

Given a Bayes Net model, we can easily write a simulator. For example, consider the CPTs given below for the simplified model with only two variables T and C.

T	P(T)
+t	0.99
-t	0.01

T	C	P(C|T)
+t	+c	0.95
+t	-c	0.05
-t	+c	0.0
-t	-c	1.0



A simple simulator in Python would be as follows:

```
import random

def get_t():
    if random.random() < 0.99:
        return True
    return False

def get_c(t):
    if t and random.random() < 0.95:
        return True
    return False

def get_sample():
    t = get_t()
    c = get_c(t)
    return [t, c]
```

We call this simple approach **prior sampling**. The downside of this approach is that it may require the generation of a very large number of samples in order to perform analysis of unlikely scenarios. If we wanted to compute $P(C|-t)$, we'd have to throw away 99% of our samples.

One way to mitigate this problem, we can modify our procedure to early reject any sample inconsistent with our evidence. For example, for the query $P(C|-t)$, we'd avoid generating a value for C unless t is false. This still means we have to throw away most of our samples, but at least the bad samples we generate take less time to create. We call this approach **rejection sampling** and it produces consistent estimates of the probabilities.

These two approaches work for the same reason, which is that any valid sample occurs with the same probability as specified in the joint PDF. In other words, the probability of every sample is based on the product of every CPT, or as I personally call it, the "every CPT participates principle".

A more exotic approach is **likelihood weighting**, which ensures that we never generate a bad sample. In this approach, we manually set all variables equal to the evidence in our query. For example, if we wanted to compute $P(C|-t)$, we'd simply declare that t is false. The problem here is that this may yield samples that are inconsistent with the correct distribution. As an example, consider the more complex four variable model for T, C, S, and E given earlier in these notes. If we wanted to compute $P(T,S,+c,+e)$, and simply picked values for T and S without taking into account the fact that c = true, and e = true, then there's no guarantee that our samples actually obey the joint PDF given by the Bayes Net. For example, if the cage only ever falls if the treasure is taken, then we'd want to ensure that T is always true instead of using the $P(T)$ distribution given in the Bayes Net.

Put differently, if we simply force some variables equal to the evidence, then our samples occur with probability given only equal to the products of the CPTs of the non-evidence variables. This means the joint PDF has no guarantee of being correct (though may be for some cases like our two variable Bayes Net). Instead, if we have sampled variables Z_1 through Z_p and fixed evidence variables E_1 through E_m a sample is given by the probability $P(Z_1...Z_p,E_1...E_m) = \prod_i^p P(Z_i|Parents(Z_i))$. What is missing is that the probability of a sample does not include all the probabilities of $P(E_i|Parents(E_i))$, i.e. not every CPT participates.

Likelihood weighting solves this issue by using a weight for each sample, which is the probability of the

evidence variables given the sampled variables. That is, instead of counting all samples equally, we can define a weight w_j for sample j that reflects how likely the observed values for the evidence variables are, given the sampled values. In this way, we ensure that every CPT participates. To do this, we iterate through each variable in the Bayes net (as we do for normal sampling), sampling a value if the variable is not an evidence variable, or changing the weight for the sample if the variable is evidence.

For example, suppose we want to calculate $P(T|+c,+e)$. For the jth sample, we'd perform the following algorithm:

- Set w_j to 1.0, and c = true and e = true.
- For T: This is not an evidence variable, so we sample t_j from $P(T)$.
- For C: This is an evidence variable, so we multiply the weight of the sample by $P(+c|t_j)$, i.e. $w_j = w_j \cdot P(+c|t_j)$.
- For S: sample s_j from $P(S \mid t_j)$.
- For E: multiply the weight of the sample by $P(+e|+c,s_j)$, i.e. $w_j = w_j \cdot P(+e|+c,s_j)$.

Then when we perform the usual counting process, we weight sample j by w_j instead of 1, where $0 <= w_j <= 1$. This approach works because in the final calculations for the probabilities, the weights effectively serve to replace the missing CPTs. In effect, we ensure that the weighted probability of each sample is given by $P(z_1...z_p,e_1...e_m) = \left[\prod_i^p P(z_i \mid Parents(z_i))\right] \cdot [\prod_i^m P(e_i \mid Parents(e_i))]$. Likelihood weighting sampling is also consistent and its pseudocode is:

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1,\ldots,X_n)$
    N, the total number of samples to be generated
  local variables: **W**, a vector of weighted counts for each value of X, initially zero

  for j = 1 to N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn, **e**)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in **x**
  return NORMALIZE(**W**)

---

function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight

  $w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
  foreach variable X_i **in** $X_1,\ldots,X_n$ **do**
    if X_i is an evidence variable with value x_i in **e**
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else **x**$[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

Figure 14.15 The likelihood-weighting algorithm for inference in Bayesian networks. In WEIGHTED-SAMPLE, each nonevidence variable is sampled according to the conditional distribution given the values already sampled for the variable's parents, while a weight is accumulated based on the likelihood for each evidence variable.

For all three of our sampling methods (prior sampling, rejection sampling, and likelihood weighting), we can get increasing amounts of accuracy by generating additional samples. However, of the three, likelihood weighting is the most computationally efficient, for reasons that are beyond the scope of this course.

Gibbs Sampling is a fourth approach for sampling. In this approach, we first set all variables to some totally random value (not taking into account any CPTs). We then repeatedly pick one variable at a time, clear its value, and resample it given the values currently assigned to all other variables. Here we can use the notion of the Markov blanket. To sample a value for a variable we can only consider the values of the variables in it's Markov blanket as it only depends on them.

For the T, C, S, E example above, we might assign t = true, c = true, s = false, and e = true. We then pick one of our four variables to resample, say S, and clear it. We then pick a new variable from the distribution $P(S|+t,+c,+e)$. This requires us knowing this conditional distribution. It turns out that we can easily compute the distribution of any single variable given all other variables. More specifically, $P(S|T,C,E)$ can be calculated only using the CPTs that connect S with its neighbors. Thus, in a typical Bayes Net, where most variables have only a small number of neighbors, we can precompute the conditional distributions for each variable given all of its neighbors in linear time. After having sampled a value for S from $P(S|+t,+c,+e)$, say $S = true$, then we choose another variable, clear its value and sample a new value for it. For example if we chose to update T then we sample a new value for T using the distribution $P(T|+c,+s)$. In the pseudocode of Gibbs sampling we sample variables sequentially. However, it should be noted that other methods of choosing the next variable to update (like choosing uniformly at random) are also widely used.

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **e**

  initialize **x** with random values for the variables in **Z**
  for j = 1 to N **do**
    for each Z_i in **Z do**
      set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

We will not prove this, but if we repeat this process enough times, our later samples will eventually converge to the correct distribution even though we may start from a low-probability assignment of values. Gibbs sampling is consistent if there are no deterministic nodes in the network. If you're curious, there are some caveats beyond the scope of the course that you can read about under the Failure Modes section of the Wikipedia article for Gibbs Sampling.

Summary

To summarize, Bayes' Nets is a powerful representation of joint probability distributions. Their topological structure encodes independence and conditional independence relationships, and we can use it to model arbitrary distributions to perform inference and sampling. We presented both exact inference and sampling inference techniques, where the latter can be employed when exact inference is infeasible.