RECITATION 8: GRAPHICAL MODELS

10-301/10-601 Introduction to Machine Learning (Summer 2023)
http://www.cs.cmu.edu/~hchai2/courses/10601
Released: July 20th, 2023
Quiz Date: July 27th, 2023
TAs: Alex, Andrew, Sofia, Tara, and Neural the Narwhal

1 HMMs

You are given the following training data:

```
win_C league_C Liverpool_D

win_C Liverpool_D league_C

Liverpool_D win_C
```

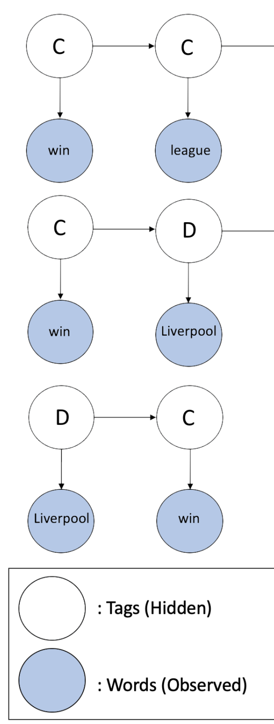


Figure 1: Visualization of Sequences

You are also given the following observed (validation) data: `Liverpool win league`

1.1 Initial, Emission, and Transition Matrices

Let each observed state $x_t \in \{1, 2, 3\}$, where 1 corresponds to `win`, 2 corresponds to `league`, and 3 corresponds to `Liverpool`. Let each hidden state $Y_t \in \{C, D\}$, where $s_1 = C$ and $s_2 = D$.

First, we need to estimate the HMM parameters - the initial probabilities: $\boldsymbol{\pi}$, the transition probability matrix: $\mathbf{B}$, and the emission probability matrix: $\mathbf{A}$. Remember that we use MLE estimation to do so:

- $\hat{C}_k = \frac{N(Y_1^{(i)}=s_k)}{N} \; \forall \, i, k$
- $\hat{B}_{jk} = \frac{N(Y_t^{(i)}=s_k, Y_{t-1}^{(i)}=s_j)}{N(Y_{t-1}^{(i)}=s_j)} \; \forall \, i, t > 1, j, k$
- $\hat{A}_{jk} = \frac{N(X_t^{(i)}=k, Y_t^{(i)}=s_j)}{N(Y_t^{(i)}=s_j)} \; \forall \, i, t, j, k$

Note: When learning an HMM, we add 1 to each count to make a pseudocount. This improves performance when evaluating unseen cases in the validation set or test set.

1. Find the initial matrix $\boldsymbol{\pi}$. Recall that $\pi_j = P(Y_1 = s_j)$.
   - Find count matrix and pseudocount matrix:

$$\begin{array}{cc} & Count \\ C & \\ D & \end{array} \xrightarrow{\text{Pseudocount}} \begin{array}{cc} & Count \\ C & \\ D & \end{array}$$

   - Normalize:

$$\boldsymbol{\pi} = \begin{array}{c} C \\ \\ D \end{array}$$

2. Find the transition matrix $\mathbf{B}$. Recall that $B_{jk} = P(Y_t = s_k \mid Y_{t-1} = s_j)$.
   - Find count matrix and pseudocount matrix:

$$\begin{array}{ccc} & C & D \\ C & & \\ D & & \end{array} \xrightarrow{\text{Pseudocount}} \begin{array}{ccc} & C & D \\ C & & \\ D & & \end{array}$$

   - Normalize:

$$\mathbf{B} = \begin{array}{ccc} & C & D \\ C & & \\ D & & \end{array}$$

3. Find the emission matrix $\mathbf{A}$. Recall that $A_{jk} = P(X_t = k \mid Y_t = s_j)$.

   - Find count matrix and pseudocount matrix:

$$\begin{array}{c|ccc} & \texttt{win} & \texttt{league} & \texttt{Liverpool} \\ C & & & \\ D & & & \end{array} \xrightarrow{\text{Pseudocount}} \begin{array}{c|ccc} & \texttt{win} & \texttt{league} & \texttt{Liverpool} \\ C & & & \\ D & & & \end{array}$$

   - Normalize:

$$\mathbf{A} = \begin{array}{c|ccc} & \texttt{win} & \texttt{league} & \texttt{Liverpool} \\ C & & & \\ D & & & \end{array}$$

1.2 The Forward Algorithm

One type of inference problem that can be answered by an HMM is **Evaluation** - computing the probability of a sequence of observations. We calculate the likelihood of observing the validation sequence:

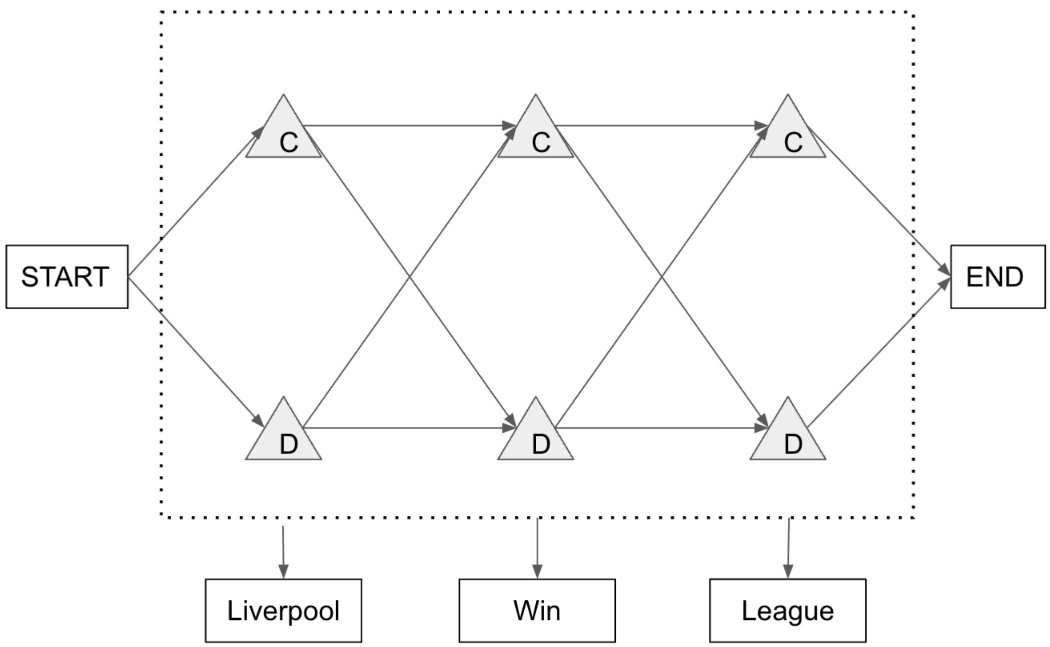
`Liverpool win league`

To do so, we calculate the forward probability matrix $\boldsymbol{\alpha}$. Recall that

$$\alpha_t(s_k) = P(x_{1:t}, Y_t = s_k)$$

We have the following bottom-up dynamic programming algorithm to calculate the forward probabilities:

```
for t = 1,...,T:
    for j = 1,...,J:
        if t == 1:
            α_1(s_j) = π_j * A_{j,x_1}
        else:
            α_t(s_j) = A_{j,x_t} * Σ_k α_{t-1}(s_k) * B_{k,j}
```



First, use the algorithm as defined above to calculate $\alpha_1(C)$ and $\alpha_1(D)$.

- $\boldsymbol{\alpha}_1 = \begin{bmatrix} \alpha_1(C) \\ \alpha_1(D) \end{bmatrix} = \begin{bmatrix} \pi_C * A_{C,x_1} \\ \pi_D * A_{C,x_1} \end{bmatrix} = \begin{bmatrix} \pi_C * A_{C,Liverpool} \\ \pi_D * A_{D,Liverpool} \end{bmatrix} =$

Observe that this can be vectorized as $\boldsymbol{\pi} \odot \boldsymbol{A}_{,\boldsymbol{x}_1}$.

Indeed, the way $\mathbf{B}$ and $\mathbf{A}$ are constructed allows us to also vectorize the computation of the forward probabilities for $1 < t \leq T$:

$$\mathbf{A}_{,x_t} \odot (\mathbf{B}^T \boldsymbol{\alpha}_{t-1})$$

- $\boldsymbol{\alpha}_2 = \begin{bmatrix} \alpha_2(C) \\ \alpha_2(D) \end{bmatrix} = \mathbf{A}_{,x_2} \odot (\mathbf{B}^T \boldsymbol{\alpha}_1) =$

- $\boldsymbol{\alpha}_3 =$

To find the likelihood of observing the validation sequence, all we need are the final forward probabilities:

$$\begin{aligned}
&P(X_1 = \texttt{Liverpool}, X_2 = \texttt{win}, X_3 = \texttt{league}) \\
&= \sum_{y_3 \in \{C,D\}} P(x_1 = \texttt{Liverpool}, x_2 = \texttt{win}, x_3 = \texttt{league}, Y_3 = y_t) \\
&= \sum_{y_t \in \{C,D\}} \alpha_3(y_t) \\
&= \\
&=
\end{aligned}$$

1.3 The Backward Algorithm

Another type of inference problem that can be answered by an HMM is computing **Marginals** - computing the marginal probability distribution for a hidden state, given a sequence of observations. Recall that

$$P(Y_t = s_k|\vec{x}) = \frac{\alpha_t(s_k)\beta_t(s_k)}{P(\vec{x})}$$

Therefore, along with the forward probability matrix $\boldsymbol{\alpha}$, we need to find the backward probability matrix $\boldsymbol{\beta}$, where

$$\beta_t(s_k) = P(x_{t+1:T}|Y_t = s_k)$$

We have a similar bottom-up dynamic programming algorithm to calculate the backward probabilities:

```
for t = T,...,0:
    for j = 1,...,k:
        if t == T:
            β_T(s_j) = 1
        else:
            β_t(s_j) = Σ_{k=1}^{J} A_{k,x_{t+1}} β_{t+1}(s_k) B_{j,k}
```

Conveniently, there is also a matrix expression for the vector of backward probabilities for a given time step $t < T$:

$$\mathbf{B}(\mathbf{A}_{,x_{t+1}} \odot \boldsymbol{\beta}_{t+1})$$

- $\boldsymbol{\beta}_3 = \begin{bmatrix} \beta_3(C) \\ \beta_3(D) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

- $\boldsymbol{\beta}_2 = \begin{bmatrix} \beta_2(C) \\ \beta_2(D) \end{bmatrix} = \mathbf{B}(\mathbf{A}_{,x_3} \odot \boldsymbol{\beta}_3) =$

- $\boldsymbol{\beta}_1 = \mathbf{B}(\mathbf{A}_{,x_2} \odot \boldsymbol{\beta}_2) =$

Now, we have our $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ matrices:

$$\boldsymbol{\alpha} = \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0.0750 & 0.2667 \\ 0.1150 & 0.0186 \\ 0.0225 & 0.0123 \end{bmatrix} \end{array}$$

$$\boldsymbol{\beta} = \begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0.0823 & 0.1072 \\ 0.2500 & 0.3229 \\ 1.0000 & 1.0000 \end{bmatrix} \end{array}$$

1. What is $P(Y_2 = C|\vec{x})$?

2. What is $P(Y_2 = D|\vec{x})$?

3. What is $P(Y_3 = C|\vec{x})$?

4. What is the minimum Bayes risk (MBR) decoder prediction for Y_2, assuming our MBR loss function is the Hamming loss?

1.4 The Viterbi Algorithm

Instead of finding the most likely hidden state at some time t, we may instead want to find the most likely sequence of hidden states. This is known as **Viterbi Decoding** - computing the most probable assignment of hidden states, given a sequence of observations.

The sequence of words you observe is again the same: `Liverpool win league`

However, you are only given the tag of the last word: `league_C`

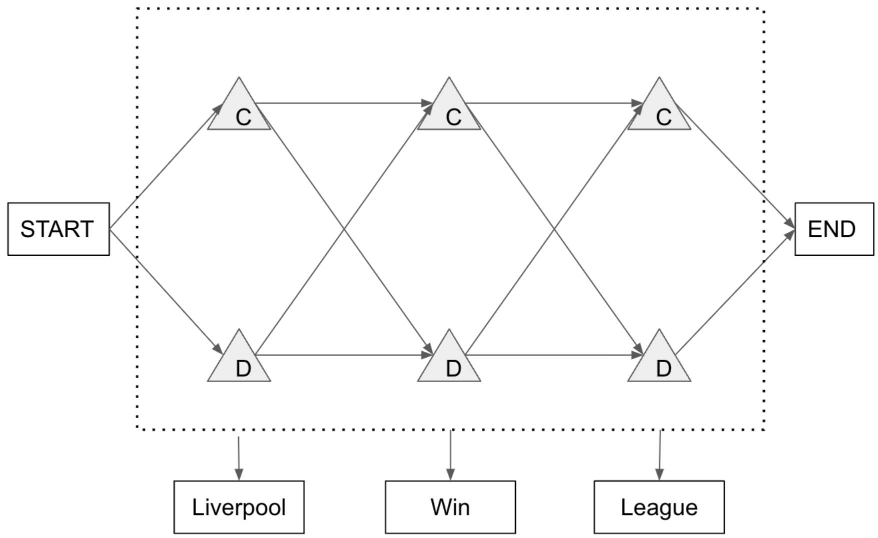
1. Recall that:

$$\omega_t(s_k) = \max_{y_{1:t-1}} P(x_{1:t}, y_{1:t-1}, y_t = s_k)$$

$$b_t(s_k) = \operatorname*{argmax}_{y_{1:t-1}} P(x_{1:t}, y_{1:t-1}, y_t = s_k)$$

   Using the formulae above and the first order Markov assumption, derive a recursive definition for $\omega_t(s_k)$ and $b_t(s_k)$ that will let you employ bottom-up dynamic programming.

Below is the trellis corresponding to the given data:



2. Annotate the trellis at the nodes that correspond to:
   (a) $\omega_1(C)$
   (b) $\omega_1(D)$
   (c) $\omega_2(C)$
   (d) $\omega_2(D)$
   (e) $\omega_3(C)$
   (f) $\omega_3(D)$

3. Find the most likely sequence of tags given the observed data:

   (a) Set up the matrices ω and b

$$\omega = \begin{array}{c} \\ \omega_0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{array} \begin{array}{c} \begin{array}{ccc} \texttt{C} & \texttt{D} & \texttt{START} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} \end{array}$$

   and

$$b = \begin{array}{c} \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{array} \begin{array}{c} \begin{array}{ccc} \texttt{C} & \texttt{D} & \texttt{END} \end{array} \\ \begin{bmatrix} - & - & - \\ - & - & - \\ - & - & - \\ - & - & - \end{bmatrix} \end{array}$$

   Initialize $w_0(\text{START}) = 1$

   (b) Solve for matrix entries using Dynamic Programming:

$$\omega_1(C) = \max_{s_j \in \text{C,D,START}} P(x_1 = \texttt{Liverpool}|Y_1 = \texttt{C})\omega_0(s_j)P(Y_1 = \texttt{C})$$
$$=$$

$$b_1(\texttt{C}) =$$

$$\omega_1(D) = \max_{s_j \in \text{C,D,START}} P(x_1 = \texttt{Liverpool}|Y_1 = \texttt{D})w_0(s_j)P(Y_1 = \texttt{D})$$
$$=$$
$$=$$

$$b_1(\texttt{D}) =$$

$$\omega_2(C) = \max_{s_j \in \text{C,D}} P(x_2 = \texttt{win}|Y_2 = \texttt{C})\omega_1(s_j)P(Y_2 = \texttt{C}|Y_1 = s_j)$$
$$= \max\Bigg(\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Bigg)$$
$$=$$

$$b_2(\texttt{C}) =$$

$$
\begin{aligned}
\omega_2(D) &= \max_{s_j \in \texttt{C},\texttt{D}} P(x_2 = \texttt{win} | Y_2 = \texttt{D}) \omega_1(s_j) P(Y_2 = \texttt{D} | Y_1 = s_j) \\
&= \max \Big(\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big) \\
&=
\end{aligned}
$$

$$
b_2(\texttt{D}) =
$$

$$
\begin{aligned}
\omega_3(C) &= \max_{s_j \in \texttt{C},\texttt{D}} P(x_3 = \texttt{league} | Y_3 = \texttt{C}) \omega_2(s_j) P(Y_3 = \texttt{C} | Y_2 = s_j) \\
&= \max \Big(\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big) \\
&=
\end{aligned}
$$

$$
b_3(\texttt{C}) =
$$

$$
\begin{aligned}
\omega_3(D) &= \max_{s_j \in \texttt{C},\texttt{D}} P(x_3 = \texttt{league} | Y_3 = \texttt{D}) \omega_2(s_j) P(Y_3 = \texttt{D} | Y_2 = s_j) \\
&= \max \Big(\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Big) \\
&=
\end{aligned}
$$

$$
b_3(\texttt{D}) =
$$

Now, to figure out the order, we set $\hat{y}_t = b_{t+1}(\hat{y}_{t+1})$

$$\hat{y}_{T+1} = \texttt{END}$$

$$\begin{aligned}\hat{y}_3 &= b_4(\texttt{END}) \\ &= \end{aligned}$$

$$\begin{aligned}\hat{y}_2 &= b_3(\qquad\quad) \\ &= \end{aligned}$$

$$\begin{aligned}\hat{y}_1 &= b_2(\qquad\quad) \\ &= \end{aligned}$$

$$\begin{aligned}\hat{y}_0 &= b_1(\qquad\quad) \\ &= \end{aligned}$$

So, the most likely sequence is:

2 Working in Log-space

2.1 Motivation

Some of the probabilities we work with in PA6 are tiny and some of them are much larger. We tend to work with the tiny ones in log-space and only get back probabilities if we really need them for some other purpose. Throughout PA6 you will keep your probabilities in log-space.In this section we will motivate why we use log-space for small values.

Given the following series of probability values:

$P(x_1 = 1)$	$P(x_2 = 1 \mid x_1 = 1)$	$P(x_3 = 1 \mid x_2 = 1, x_1 = 1)$
0.002	0.004	0.003

We want to find $P(x_1 = 1, x_2 = 1, x_3 = 1)$. Suppose we have a calculator which only has 4 decimal places of precision, so it can only store values of format X.XXXX

1. What is the correct value of $P(x_1 = 1, x_2 = 1, x_3 = 1)$ without any precision limits?

   $P(x_1 = 1, x_2 = 1, x_3 = 1) = P(x_3 = 1 \mid x_2 = 1, x_1 = 1) * P(x_2 = 1 \mid x_1 = 1) * P(x_1 = 1)$
   $=$

2. What is the value of $P(x_1 = 1, x_2 = 1, x_3 = 1)$ using our faulty calculator?

   $P(x_1 = 1, x_2 = 1)$

   $= P(x_2 = 1 \mid x_1 = 1)P(x_1 = 1) =$

   $P(x_1 = 1, x_2 = 1, x_3 = 1) =$

3. How do the values of $P(x_1 = 1, x_2 = 1, x_3 = 1)$ from part (1) and (2) compare?

   No precision limits: $P(x_1 = 1, x_2 = 1, x_3 = 1) =$

   Faulty calculator: $P(x_1 = 1, x_2 = 1, x_3 = 1) =$

4. What is the value of $P(x_1 = 1, x_2 = 1, x_3 = 1)$ if we perform the same computation but in log space?

   $\log\big(P(x_1 = 1, x_2 = 1, x_3 = 1)\big)$

   $= \log(x_1 = 1) + \log(P(x_2 = 1 \mid x_1 = 1)) + \log(P(x_3 = 1 \mid x_2 = 1, x_1 = 1)$

   $=$

   If we were to recover our value of $P(x_1 = 1, x_2 = 1, x_3 = 1) = e^{\log\big(P(x_1=1,x_2=1,x_3=1)\big)} =$

   This is good! But we can use the log sum exp trick to extend its use to even smaller scales.

2.2 Forward and Backward Algorithm in Log Space

In the forward algorithm, recall that the entries in $\boldsymbol{\alpha}$ can be computed using the bottom-up dynamic programming algorithm:

- $\alpha_1(j) = \pi_j A_{jx_1}$
- For $t > 1$, $\alpha_t(j) = A_{jx_t} \sum_{k=1}^{J} \alpha_{t-1}(k) B_{kj}$

1. Derive $\log\big(\alpha_1(j)\big)$ in terms of $\log(\pi_j)$ and $\log(A_{jx_1})$

   $\log\big(\alpha_1(j)\big) = \log\big(\pi_j A_{jx_1}\big) =$

2. Derive $\log\big(\alpha_t(j)\big)$ in terms of $\log\big(\alpha_{t-1}(k)\big)$ and $\log A_{kj}$

   $\log\big(\alpha_t(j)\big)$
   $= \log\big(A_{jx_t} \sum_{k=1}^{J} \alpha_{t-1}(k) B_{kj}\big)$
   $= \log(A_{jx_t}) + \log\big(\sum_{k=1}^{J} \alpha_{t-1}(k) B_{kj}\big)$

   $=$

In the backward algorithm, we also have a similar bottom-up dynamic programming algorithm:

- $\beta_T(j) = 1$
- For $1 \leq t \leq T-1$, $\beta_t(j) = \sum_{k=1}^{J} A_{kx_{t+1}} \beta_{t+1}(k) B_{jk}$

1. Derive $\log\big(\beta_T(j)\big)$

   $\log\big(\beta_T(j)\big) = \log(1) = 0$

2. Derive $\log\big(\beta_t(j)\big)$ in terms of $\log(A_{kx_{t+1}})$, $\log\big(\beta_{t+1}(k)\big)$, and $\log(B_{jk})$

   $\log\big(\beta_t(j)\big)$
   $= \log\big(\sum_{k=1}^{J} A_{kx_{t+1}} \beta_{t+1}(k) B_{jk}\big)$
   $= \log\left(\sum_{k=1}^{J} e^{\log\left(A_{kx_{t+1}} \beta_{t+1}(k) B_{jk}\right)}\right)$

   $=$

After transforming the equations into log form, you may discover calculations of the following type:

$$\log \sum_i \exp\left(v_i\right)$$

This may be programmed as is, but $\exp\left(v_i\right)$ may cause underflow when v_i is large and negative. One way to avoid this is to use the log-sum-exp trick.

The log-sum-exp trick simply adds the maximum value in the vector to the log probabilities as follows:

$$m + \log \sum_i \exp(v_i - m))$$

where

$$m = \max_i(v_i)$$

3 Bayesian Networks

3.1 HMMs as Bayes Nets

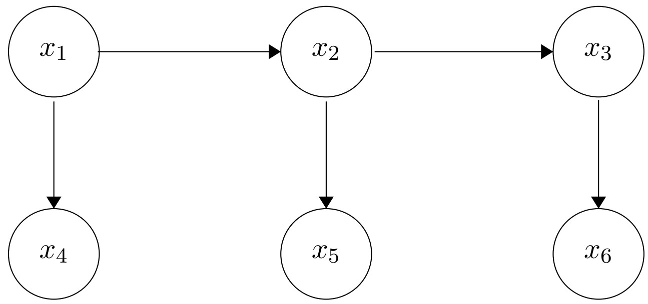


Figure 2: Graphical Model

1. Write down the factorization of the above directed graphical model.

   $P(X_1, X_2, X_3, X_4, X_5, X_6)$

2. Given X_3, what are the relationships (cond. independent or not) between the random variables listed below

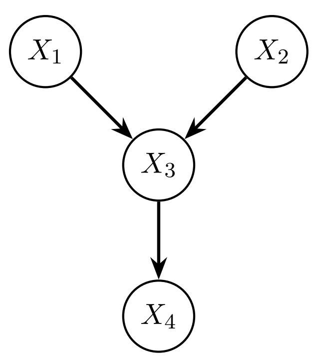
   - $(X_1 \_\_\_\_\_\_ X_4)|X_3$
   - $(X_1 \_\_\_\_\_\_ X_2)|X_3$
   - $(X_4 \_\_\_\_\_\_ X_6)|X_3$

3. Given the graph structure and assuming all variables are boolean valued, how many parameters are required to learn the graphical model?

4. Without the Bayesian network, how many parameters are required to learn the joint probability model of these five random variables?

3.2 Conditional Independence

Consider the graphical model below over 4 boolean random variables:



We also have the associated conditional probability tables (as an example, the top left element of the bottom table reads as $P(X_4 = 0|X_3 = 0) = 0.8$):

$X_1 = 0$	0.3
$X_1 = 1$	0.7

$X_2 = 0$	0.5
$X_2 = 1$	0.5

	$X_1 = 0, X_2 = 0$	$X_1 = 0, X_2 = 1$	$X_1 = 1, X_2 = 0$	$X_1 = 1, X_2 = 1$
$X_3 = 0$	0.4	0.7	0.8	0.5
$X_3 = 1$	0.6	0.3	0.2	0.5

	$X_3 = 0$	$X_3 = 1$
$X_4 = 0$	0.8	0.25
$X_4 = 1$	0.2	0.75

Table 1: Conditional Probability tables

For the following questions, indicate whether the independence claim is true or false.

1. $(X_1 \perp\!\!\!\perp X_2) \mid X_3$

   ◯ True

   ◯ False

2. $(X_1 \perp\!\!\!\perp X_4) \mid X_3$

   ◯ True

   ◯ False

Based on the graphical model and the conditional probability tables, calculate the following values:

1. What is $P(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0)$?

2. What is $P(X_1 = 1, X_2 = 1, X_4 = 1)$?

3. What is $P(X_2 = 1|X_4 = 1, X_3 = 0)$?

4 Dynamic Programming

DP Notebook

To access this Colab notebook you will need to be logged into Google Drive with your Andrew email.