Local Probability Models

Representation

Professor Ajoodha

Lecture 3
Representation

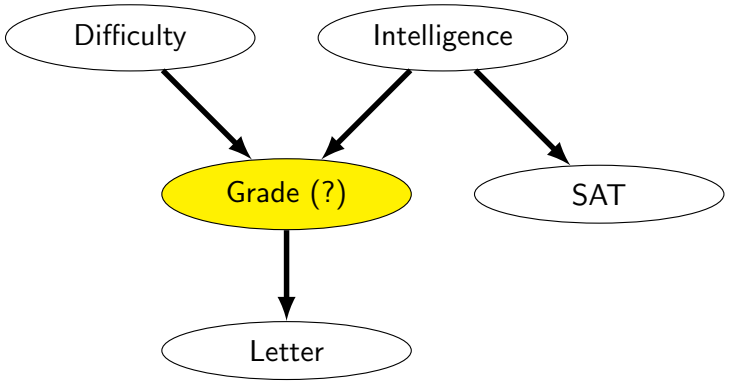
School of Computer Science and Applied Mathematics
The University of the Witwatersrand, Johannesburg

ExplainableAI Lab
— MODELLING. DECISION MAKING. CAUSALITY —

Problem Statement

1. **Independence**: allows us to factorise complex distributions in to a product of factors
2. But, how can these factors be represented?

   CPDs: Conditional Probability Distributions
3. In otherwords, given a graphical model:



How can Grade be represented?

Tabular CPDs

When dealing with discrete random variables we can use tabular representation of CPDs (Tabular CPDs or CPTs)

Where $P(X|Pa_X)$ is a table that contains an entry for each **joint assignment** to X and Pa_X.

For the table to be a legal distribution we must have that:

1. all values in the table are non-negative;
2. $\sum_{x \in Val(X)} P(x|Pa_x) = 1$.

Example:

Grade	Difficulty	Intelligence	$P(\texttt{Grade} \mid \texttt{Difficulty, Intelligence})$
F	F	F	0.1
F	F	T	0.2
F	T	F	0.09
F	T	T	0.5
T	F	F	0.01
T	F	T	0.033
T	T	F	0.033
T	T	T	0.034

Tabular CPDs

Conditional Probability Tables (CPTs) are a inherent part of the Bayesian Network Representation

However, Limitations include:

1. Random variables with infinite domains (continuous variables) can not be stored in a table.
2. Table size $= |Val(Pa_X)|.|Val(X)|$
3. Table size grows exponentially with the number of parents
4. Example: 2 parents $(+$ X$)$ we need $= 8$ values.
5. Example: 6 parents $(+$ X$)$ we need $= 128$ values.
6. What about obtaining the probabilities for 128 values in a table? Our expert will lose patience.

A CPD just needs to specify a conditional probability. However, how this is done is open for interpretation.

Deterministic CPDs

The simplest type of non-tabula CPD is a function f. That is, $f : Val(Pa_X) \mapsto Val(X)$ such that:

$$P(x \mid pa_x) = \begin{cases} 1 & \text{if } x = f(pa_X) \\ 0 & \text{if } x \texttt{ otherwise} \end{cases}$$

For example: If X is continuous, then $P(X \mid Y, Z)$ is modelled as the deterministic function: $X = Y + Z$.

This takes much less space in a computer than a CPT.

Can also use $X = 2Y + -3Z + 1$ or $X = \sin y + e^z$

Deterministic CPDs

Deterministic CPDs are also useful in modelling domains

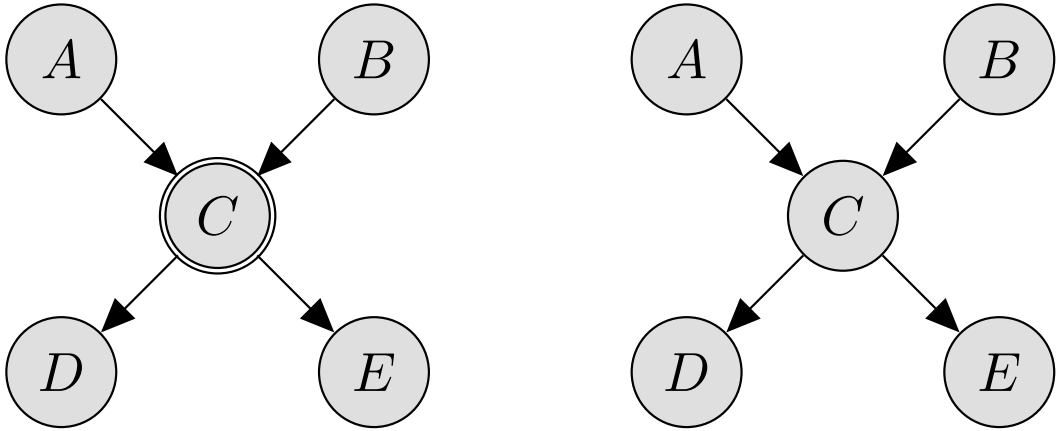
Example 1: Using a deterministic OR in a circuit to detect faults.

Example 2: The blood type of a person for two alleles, G_1 and G_2, can be modelled as:

$$\texttt{Blood type} = \begin{cases} ab & \text{if } G_1 = a \texttt{ OR } G_2 = b \\ a & \text{if } (G_1 = a) \texttt{ AND } (G_2 = a \texttt{ OR } G_2 = o) \\ b & \text{if } (G_1 = b) \texttt{ AND } (G_2 = b \texttt{ OR } G_2 = o) \\ o & \text{if } G_1 = o \texttt{ AND } G_2 = o \end{cases}$$

Independencies

Suppose we have the following two graphical models, where C is a deterministic in the left model:



Since C is deterministic in the left model:

If we know A and B, then we also know C.

Therefore, $(D \perp E \mid A, B)$.

However, in the right model:

If we know A and B, but not C.

Then this does not necessary hold: $(D \perp E \mid A, B)$.

Deterministic Separation is a simple extension to the d-separation algorithm we did previously.

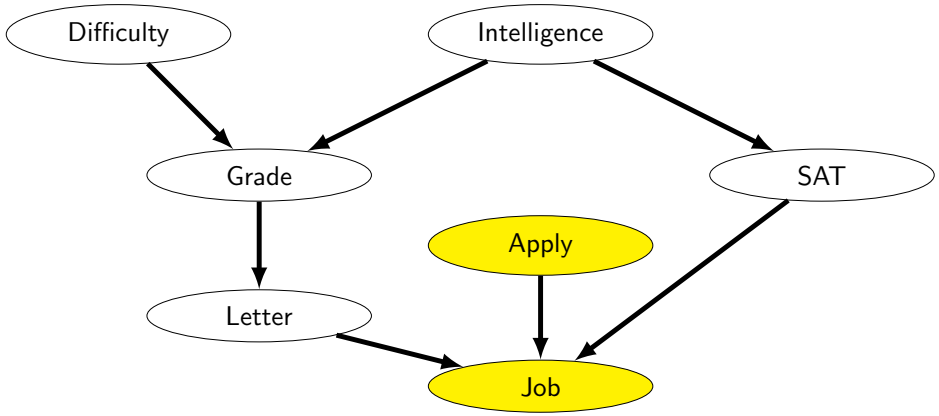
Deterministic Separation

Deterministic Separation is a simple extension to the d-separation algorithm we did previously.

> When we construct $\mathbf{Z}$ we just need to include whatever deterministic variables as evidence in $\mathbf{Z}$.

Algorithm on Page 160 of the textbook.

Context-Specific CPDs

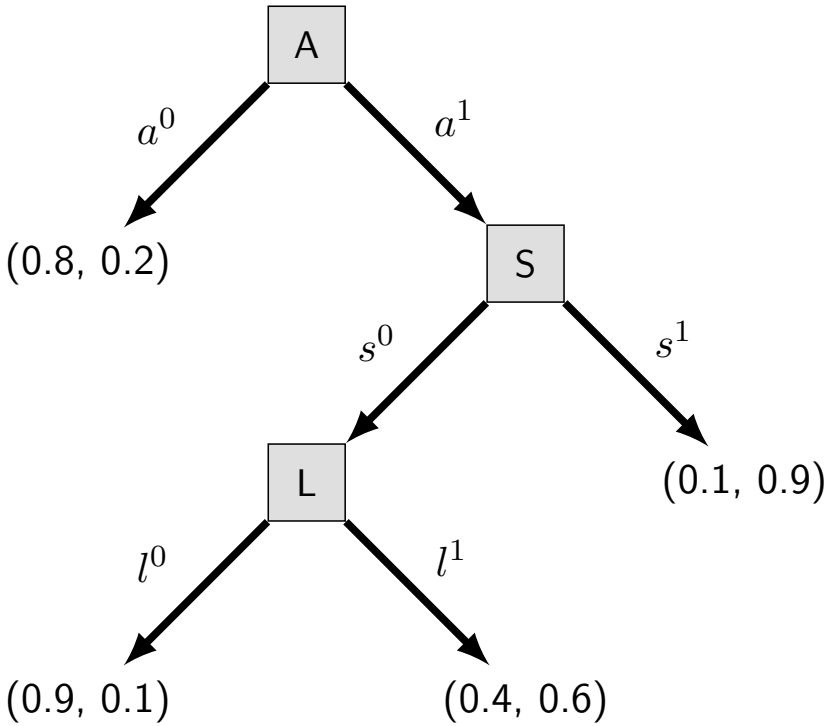


There are some context specific independence that we can observe:

- $(J \perp_c S, L \mid a^0)$ since $P(J \mid a^0, s, l)$ is the same $\forall\ s, l$.
- $(J \perp_c L \mid s^1, a^1)$ since $P(J \mid a^0, s^1, l)$ is the same $\forall\ l$.

a^0	s^0	l^0	j^0	0.1
a^0	s^0	l^0	j^1	0.025
a^0	s^0	l^1	j^0	0.1
a^0	s^0	l^1	j^1	0.025
a^0	s^1	l^0	j^0	0.1
a^0	s^1	l^0	j^1	0.025
a^0	s^1	l^1	j^0	0.1
a^0	s^1	l^1	j^1	0.025
a^1	s^0	l^0	j^0	0.1125
a^1	s^0	l^0	j^1	0.0125
a^1	s^0	l^1	j^0	0.05
a^1	s^0	l^1	j^1	0.075
a^1	s^1	l^0	j^0	0.0125
a^1	s^1	l^0	j^1	0.1125
a^1	s^1	l^1	j^0	0.0125
a^1	s^1	l^1	j^1	0.1125

Tree-CPDs



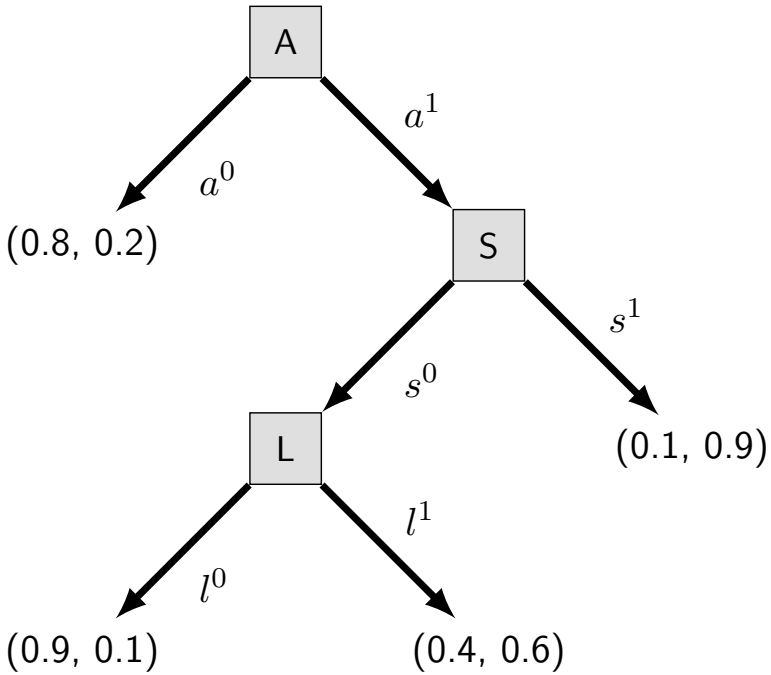
The tree codes 4 parameters instead of 16 in the table.

a^0	s^0	l^0	j^0	0.1
a^0	s^0	l^0	j^1	0.025
a^0	s^0	l^1	j^0	0.1
a^0	s^0	l^1	j^1	0.025
a^0	s^1	l^0	j^0	0.1
a^0	s^1	l^0	j^1	0.025
a^0	s^1	l^1	j^0	0.1
a^0	s^1	l^1	j^1	0.025
a^1	s^0	l^0	j^0	0.1125
a^1	s^0	l^0	j^1	0.0125
a^1	s^0	l^1	j^0	0.05
a^1	s^0	l^1	j^1	0.075
a^1	s^1	l^0	j^0	0.0125
a^1	s^1	l^0	j^1	0.1125
a^1	s^1	l^1	j^0	0.0125
a^1	s^1	l^1	j^1	0.1125

Rule-CPDs

A rule $\rho = \langle \mathbf{c}, p \rangle$ where $\mathbf{c}$ is an assignment to some subset of variables $\mathbf{C}$, and $p \in [0, 1]$.

$$\rho_1 = \langle a^0, j^0; 0.8 \rangle$$
$$\rho_2 = \langle a^0, j^1; 0.2 \rangle$$
$$\rho_3 = \langle a^1, s^0, l^0, j^0; 0.9 \rangle$$
$$\rho_4 = \langle a^1, s^0, l^0, j^1; 0.1 \rangle$$
$$\rho_5 = \langle a^1, s^0, l^1, j^0; 0.4 \rangle$$
$$\rho_6 = \langle a^1, s^0, l^1, j^1; 0.6 \rangle$$
$$\rho_7 = \langle a^1, s^1, j^0; 0.1 \rangle$$
$$\rho_8 = \langle a^1, s^1, j^1; 0.9 \rangle$$



Tree-CPDs vs Rule CPDs

Trees can capture the dependency structure in one data structure.

Rule CPDs capture the dependency structure into finer grained representation.

Rule CPDs are **not only** simple transformation of tree-CPDs.

Not every rule-based CPD can be captured compactly as a tree. (See page 168)

There are also other representations of Context-Specific CPDs:

- Multinets
- Similarity Networks

Template-based Models

Representation

Professor Ajoodha

School of Computer Science and Applied Mathematics
The University of the Witwatersrand, Johannesburg

Problem Statement

PGMs specify a joint distribution over a set $\mathcal{X}$ of random variables.

There are two cases where this representation is inadequate since many domains specify complex spaces than can be described in a fixed set $\mathcal{X}$.

Case 1: In temporal settings where we want to specify distributions over systems whose state changes over time.

In these examples the trajectory is of varying length or infinite.

- A patient in ICU (Medical)
- Tracking a robots location in an environment (RL)

Case 2: Where our $\mathcal{X}$ needs to specify an entire set of objects each with their own properties.

- Family trees in Genetics

Problem Statement

It would be much easier to specify a template for a class of distributions.

In this lecture you will learn how to specify distributions over richly structured spaces, consisting of multiple objects, interelated in a variety of ways.

1. Dynamic Bayesian networks is a single compact model which captures the properties of system dynamics (over trajectories).
2. Where we have multiple objects that are related we specify template models to generally describe a distribution

Template Models Notation

We turn our attention to dynamic settings

Here we want to reason about the state of the world as it evolves over time

In otherwords, we need to retrieve a **system state** whose value at time t is a model over relevant attributes (about $\mathcal{X}$) at time t.

The system state is modelled w.r.t a variables set $\mathcal{X}$

$X_i^{(t)}$ denotes a variable X_i at time t.

We refer to X_i as a template variable

$\mathbf{X}^{(t_1:t_2)}(t_1 < t_2)$ is a set of variables $\{\mathbf{X}^{(t)} : t \in [t_1, t_2]\}$

Assumption 1: Time Granularity

Firstly, we discretize the timeline into time slices.

time slices are measurements of the system state taken at intervals that are regularly spaced with a predetermined time granularity ($\triangle$)

From this assumption we achieve $\mathcal{X}^{(0)}, \mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(T)}$, where $\mathcal{X}^{(t)}$ are the random variables that represent the system state at time t.

From this assumption we can model the joint probability distribution as:

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

The distribution over all trajectories is the product of conditional distributions, for the variables in each time slice given the precedings ones

Assumption 2: Markov Assumption

Secondly, we simplify by assuming that **the future** is conditionally independent of **the past** given **the present**.
These systems are called Markovian.
The Markov assumption simplifies our distribution considerably:

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

Is this a reasonable assumption? (Think about a robot moving in an environment or driving a car on the highway).
Usually if we use a rich enough system state then the Markov assumption is a reasonable assumption. (Use informative variables which capture characteristic of the trajectory.)
You can also use a semi-Markov assumption.

Assumption 3: Stationarity/Time Invariance/Homogeneous

Finally, we make the Stationarity assumption.

This assumption is crucial for the template based representation.

We assume that $P(\mathcal{X}' \mid \mathcal{X})$ is the same for all t.

We can therefore represent the distribution as a transition model:

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

Two-time-slice Bayesian Network

With these assumptions we can represent the distribution over infinite trajectories compactly.

That is we only need to define two things:

1. The initial state distribution
2. and the transition model $P(\mathcal{X}' \mid \mathcal{X})$

We can thus define a two-time-slice Bayesian network (2-TBN) for a process over $\mathcal{X}$ as:

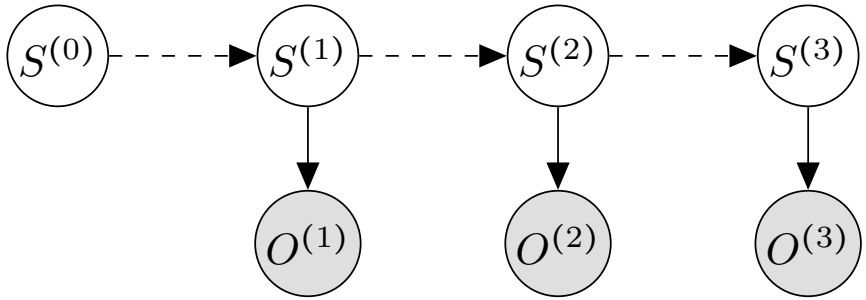
- a conditional Bayesian network over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of interface variables.

Interface variables $(\mathcal{X}_I)$ are those who have a direct effect on variables at $t+1$.

The 2-TBN models the following distribution:

$$P(\mathcal{X}' \mid \mathcal{X}) = P(\mathcal{X}' \mid \mathcal{X}_I) = \prod_{i=1}^{n} P(X_i' \mid Pa_{X_i'})$$

Structural Definitions



There are some terminology that you need to remember:

1. **Template factor**: $P(S^{(2)} \mid S^{(1)})$.
2. **Inter-time-slice edge**: connects variables between time slices (dotted arrow). Influence between variables is greater than $\triangle$.
3. **Intra-time-slice edge**: connects variables within the same time slice (solid arrow). Influence between variables is shorter than $\triangle$.
4. **Persistence edge**: Variables persist with high probability. (e.g. $S^{(1)} \rightarrow S^{(2)}$)
5. **Persistence variables**: Variables connected with persistent edges (e.g. $S^{(1)}$)

Dynamic Bayesian Network

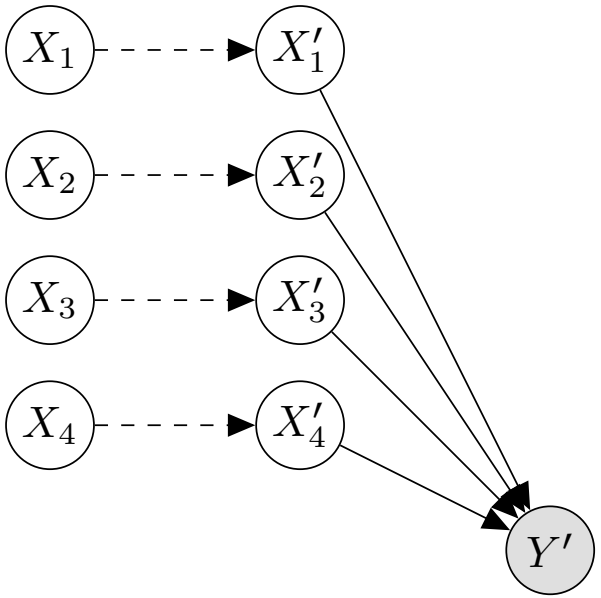
Once we define a 2-TBN it is very easy to extend it to a dynamic Bayesian network (DBN).

A dynamic Bayesian network is a pair $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$

- where $\mathcal{B}_0$ is a Bayesian network over $\mathcal{X}^{(0)}$ representing the initial distribution.
- and $\mathcal{B}_{\rightarrow}$ is a 2-TBN for the process.

You can simply unroll the network as a distribution over $\mathcal{X}^{(0:T)}$.

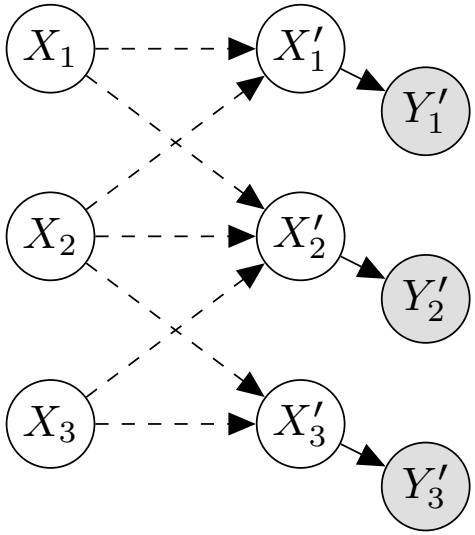
Example 1: Factorial HMM



This is called a **factorial HMM**.

To distinguish several sounds playing on a single microphone.

Example 1: Coupled HMM



This is called a **coupled HMM**.

Can be used to monitor the temperate in a building over time (for example fire alarms).

In addition to monitoring the temperature in a single room, this model can also capture the interaction between the hidden states in adjacent rooms.

State-Observation Models

Another way to representing a process is using a **State-Observation Model**.

We generally use these models when we have access to the system state through noise senors.

So we separate the dynamics of the system from our ability to sense it.

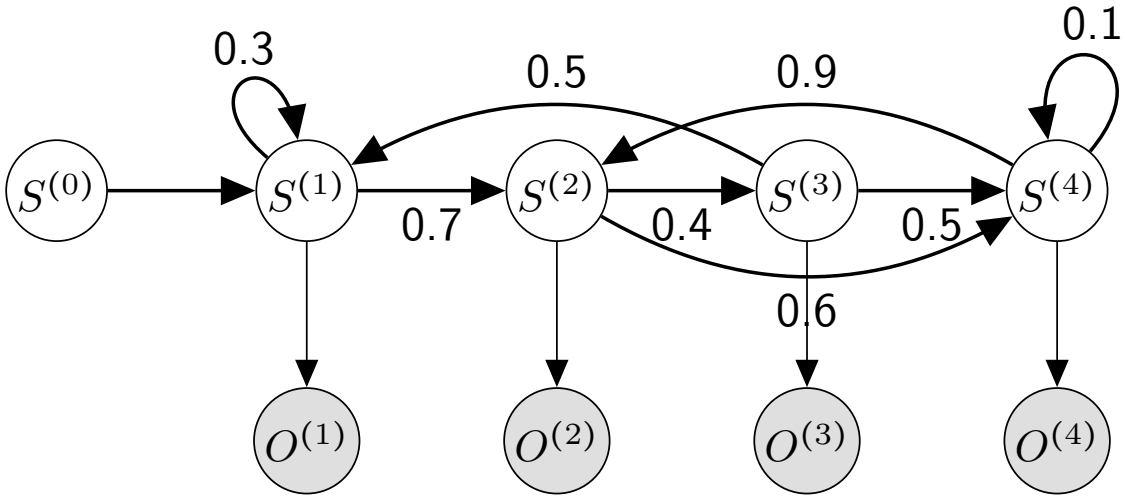
State observation models are made-up of:

1. Transition Model: $P(\mathbf{X}' \mid \mathbf{X})$
2. Observation Model: $P(\mathbf{O} \mid \mathbf{X})$

Two important architectures:

1. Hidden Markov models (Speech recognition, Robot Localisation)
2. Linear dynamical systems (GPS, tracking aeroplanes with noisy measurements)

Hidden Markov Models



A HMM is the simplest state-observation model

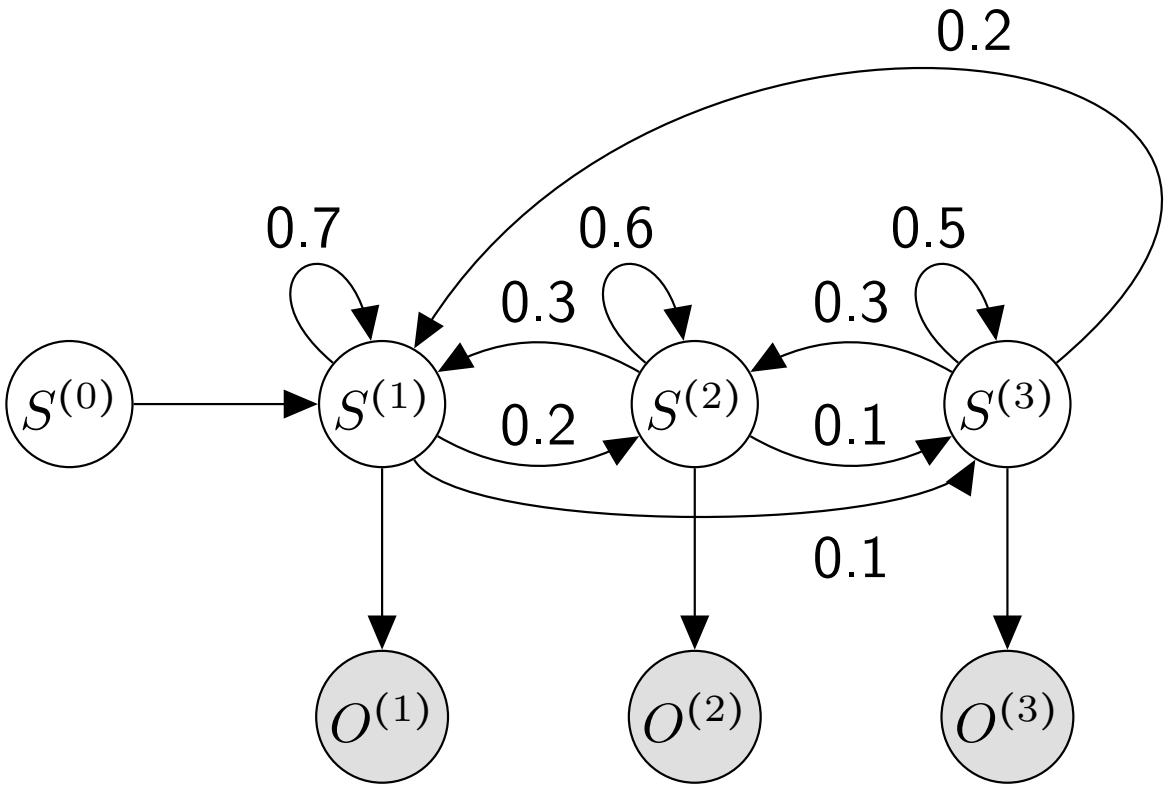
The transition model $P(S' \mid S)$ in an HMM is sparse:

	s_1	s_2	s_3	s_4
s_1	0.3	0.7	0	0
s_2	0	0	0.4	0.6
s_3	0.5	0	0	0.5
s_4	0	0.9	0	0.1

Calculating the probability of the sequence

- What is the probability of oberving the sequence $O = \{0, 1, 2\}$ given the below HMM parameters, denoted λ?
  1. Number hidden states $= 3$
  2. Number Observable symbols $= 4$
  3. Initial state probabilities: $\pi_1 = 0.6, \pi_2 = 0.2, \pi_3 = 0.2$
  4. Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
  5. Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$

Step 0: Visualisation



Step 1: Calculate Initial State Probabilities with first symbol (t = 1) for $O = \{0, 1, 2\}$

- We need to calculate the probability of starting in each of the possible hidden states for $O = \{0, 1, 2\}$.
- Compute **forward variable**: $\alpha_t(i)$: the probability of being in hidden state i at time t given the observed sequence up to time t.
- **Recall**: $\pi_1 = 0.6, \pi_2 = 0.2, \pi_3 = 0.2$
- **Recall**: Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- **Then**:
  1. $\alpha_1(1) = \pi_i b_{1,0} = 0.6 * 0.2 = 0.12$
  2. $\alpha_1(2) = \pi_i b_{2,0} = 0.2 * 0.3 = 0.06$
  3. $\alpha_1(3) = \pi_i b_{3,0} = 0.2 * 0.3 = 0.06$

Induction Step 2: Calculate (t = 2) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_1(1) = 0.12$; $\alpha_1(2) = 0.06$; $\alpha_1(3) = 0.06$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_2(1) &= \sum_{i=1}^{3} \alpha_1(i) a_{i,1} b_{1,1} \\ &= \alpha_1(1) a_{1,1} b_{1,1} + \alpha_1(2) a_{2,1} b_{1,1} + \alpha_1(3) a_{3,1} b_{1,1} \\ &= \left(0.12 \times 0.7 \times 0.3\right) + \left(0.06 \times 0.3 \times 0.3\right) + \left(0.06 \times 0.2 \times 0.3\right) \\ &= 0.0342 \end{aligned}$$

Induction Step 2: Calculate (t = 2) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_1(1) = 0.12$; $\alpha_1(2) = 0.06$; $\alpha_1(3) = 0.06$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_2(2) &= \sum_{i=1}^{3} \alpha_1(i) a_{i,2} b_{2,1} \\ &= \alpha_1(1) a_{1,2} b_{2,1} + \alpha_1(2) a_{2,2} b_{2,1} + \alpha_1(3) a_{3,2} b_{2,1} \\ &= \left(0.12 \times 0.2 \times 0.2\right) + \left(0.06 \times 0.6 \times 0.2\right) + \left(0.06 \times 0.3 \times 0.2\right) \\ &= 0.0156 \end{aligned}$$

Induction Step 2: Calculate (t = 2) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_1(1) = 0.12$; $\alpha_1(2) = 0.06$; $\alpha_1(3) = 0.06$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_2(3) &= \sum_{i=1}^{3} \alpha_1(i) a_{i,3} b_{3,1} \\ &= \alpha_1(1) a_{1,3} b_{3,1} + \alpha_1(2) a_{2,3} b_{3,1} + \alpha_1(3) a_{3,3} b_{3,1} \\ &= \left(0.12 \times 0.1 \times 0.3\right) + \left(0.06 \times 0.1 \times 0.3\right) + \left(0.06 \times 0.5 \times 0.3\right) \\ &= 0.0144 \end{aligned}$$

Induction Step 3: Calculate (t = 3) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_2(1) = 0.0342$; $\alpha_2(2) = 0.0156$; $\alpha_2(3) = 0.0144$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_3(1) &= \sum_{i=1}^{3} \alpha_2(i) a_{i,1} b_{1,2} \\ &= \alpha_2(1) a_{1,1} b_{1,2} + \alpha_2(2) a_{2,1} b_{1,2} + \alpha_2(3) a_{3,1} b_{1,2} \\ &= \left(0.0342 \times 0.7 \times 0.3\right) + \left(0.0156 \times 0.3 \times 0.3\right) + \left(0.0144 \times 0.2 \times 0.3\right) \\ &= 0.00945 \end{aligned}$$

Induction Step 3: Calculate (t = 3) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_2(1) = 0.0342$; $\alpha_2(2) = 0.0156$; $\alpha_2(3) = 0.0144$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_3(2) &= \sum_{i=1}^{3} \alpha_2(i) a_{i,2} b_{2,2} \\ &= \alpha_2(1) a_{1,2} b_{2,2} + \alpha_2(2) a_{2,2} b_{2,2} + \alpha_2(3) a_{3,2} b_{2,2} \\ &= \left(0.0342 \times 0.2 \times 0.2\right) + \left(0.0156 \times 0.6 \times 0.2\right) + \left(0.0144 \times 0.3 \times 0.2\right) \\ &= 0.004104 \end{aligned}$$

Induction Step 3: Calculate (t = 3) for $O = \{0, 1, 2\}$

- **Recall**: $\alpha_2(1) = 0.0342$; $\alpha_2(2) = 0.0156$; $\alpha_2(3) = 0.0144$
- **Recall** Transition probability matrix: $\begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.2 & 0.3 & 0.5 \end{pmatrix}$
- **Recall** Observation model: $\begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.2 \\ 0.3 & 0.2 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.2 & 0.2 \end{pmatrix}$
- Now we need to use the initialisation step to workout:

$$\begin{aligned} \alpha_3(3) &= \sum_{i=1}^{3} \alpha_2(i) a_{i,3} b_{3,2} \\ &= \alpha_2(1) a_{1,3} b_{3,2} + \alpha_2(2) a_{2,3} b_{3,2} + \alpha_2(3) a_{3,3} b_{3,2} \\ &= \left(0.0342 \times 0.1 \times 0.2\right) + \left(0.0156 \times 0.1 \times 0.2\right) + \left(0.0144 \times 0.5 \times 0.2\right) \\ &= 0.004104 \end{aligned}$$

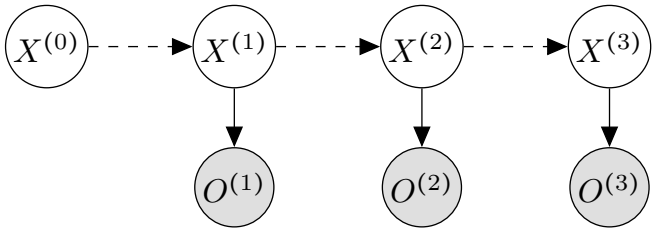
Termination Step 4: $P(O \mid \lambda)$ for $O = \{0, 1, 2\}$

- Now we simply sum the forward variables from the last step.
- λ is set of model parameters which define the HMM.

$$\begin{aligned} P(O \mid \lambda) &= P(0, 1, 2 \mid \lambda) \\ &= \alpha_3(1) + \alpha_3(2) + \alpha_3(3) \\ &= 0.00945 + 0.004104 + 0.004104 \\ &= 0.017658 \end{aligned}$$

- Therefore, the probability of observing the sequence $O = 0, 1, 2$ given λ is 0.017658.

Linear Dynamical Systems (Kalman Filter)



Evolves Linearly over time with some Gaussian noise

Continuous variables with linear Gaussian dependencies.

Transition and observation model is as follows:

$$P(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}) = \mathcal{N}(A\mathbf{X}^{(t-1)}; Q)$$
$$P(O^{(t)} \mid \mathbf{X}^{(t)}) = \mathcal{N}(H\mathbf{X}^{(t)}; R)$$

$\mathbf{X}$ (n vector) is state variables

$\mathbf{O}$ (m vector) is observation variables

A $(n \times n)$ is the linear transition model

H $(n \times m)$ is the linear observation model

Q $(n \times n)$ is Gaussian noise for system dynamics;

R $(m \times m)$ is Gaussian noise for observations.

Plate Models

We will now define plate models for directed PGMs.

This framework is especially used in Object Relational Domains

That is, where objects, classes and inheritance are directly supported in database schemas and in the query language base.

"Plates" since objects in a plate contain the same set of attributes.

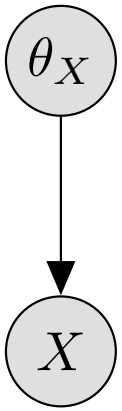
In this framework we include the CPD parameters directly in the model.

This allows us to explicitly indicate that variables are sampled from the same CPD.

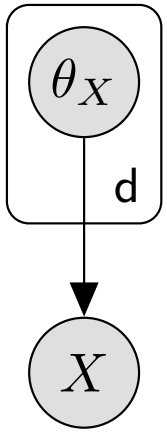
We can also overlap plates.

Plate Models

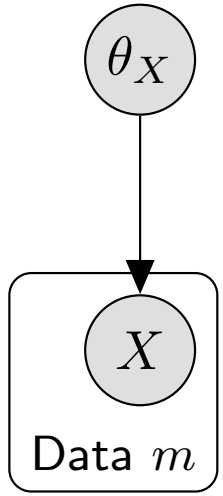
What is the difference between the following?



- The same distribution generates a single random variable.



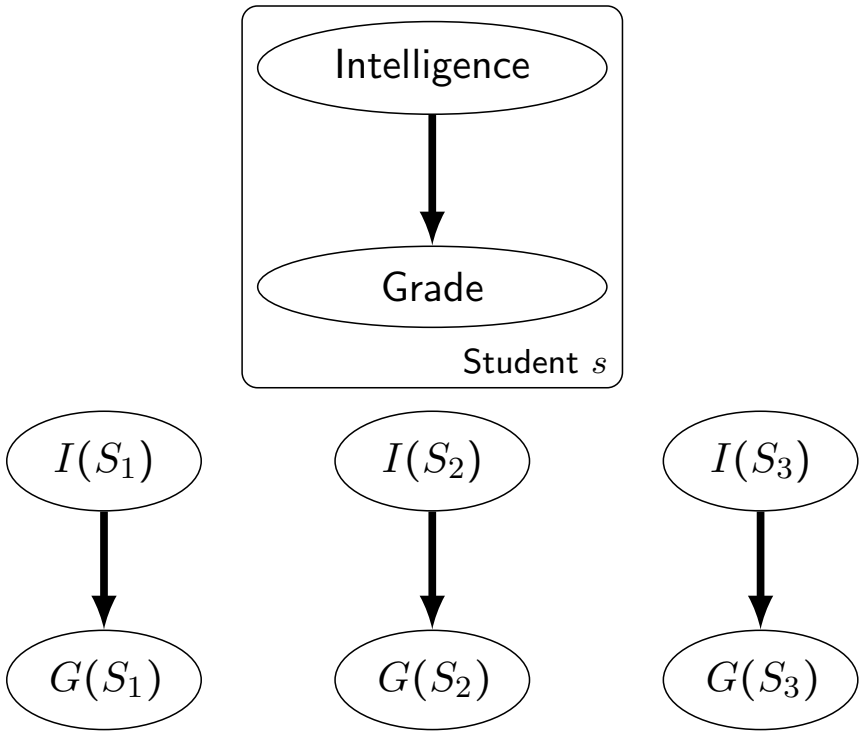
- Each data value for X comes from multiple different distributions (Each coin toss is sampled from a different coin)



- Multiple random variables generated from the same distribution.

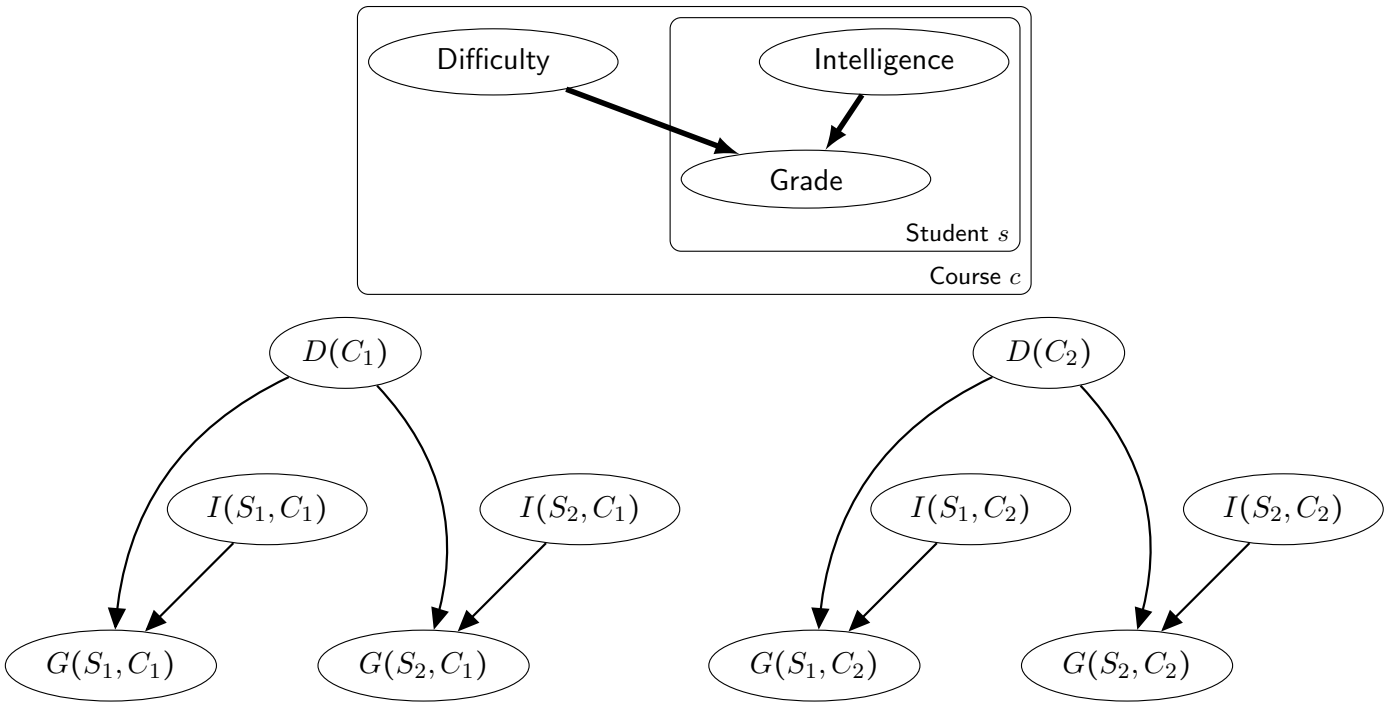
Example 1

Suppose we want to model the intelligence and grade of each (individual) student at the university.



Example 2

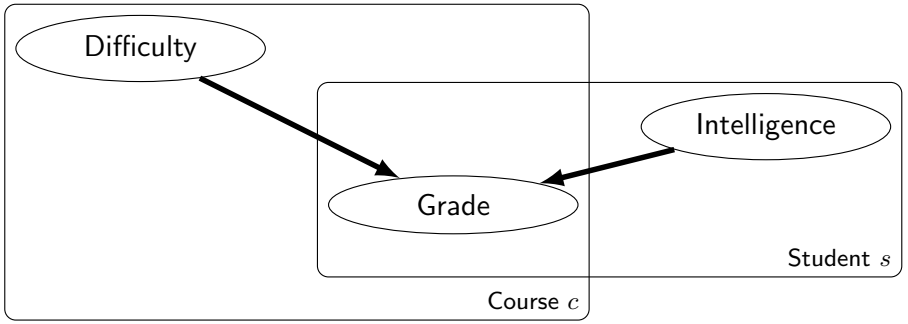
A course has multiple students, each with their own grade and this grade depends on the difficulty of the course.

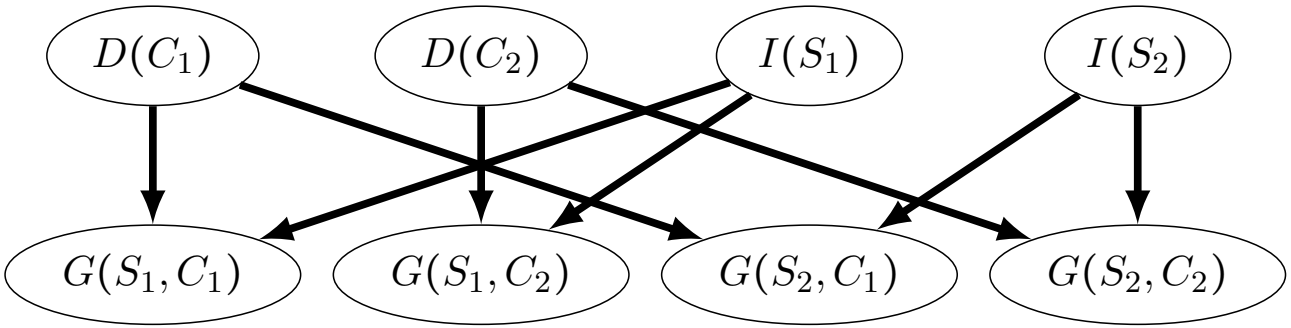


The Intelligence is not supposed to be associated with the course and grade pair!

Example 3

Let us now make the intelligence variable independent of the course, by taking it out of the course plate.





Explicit parameters

Do not forget the parameters!

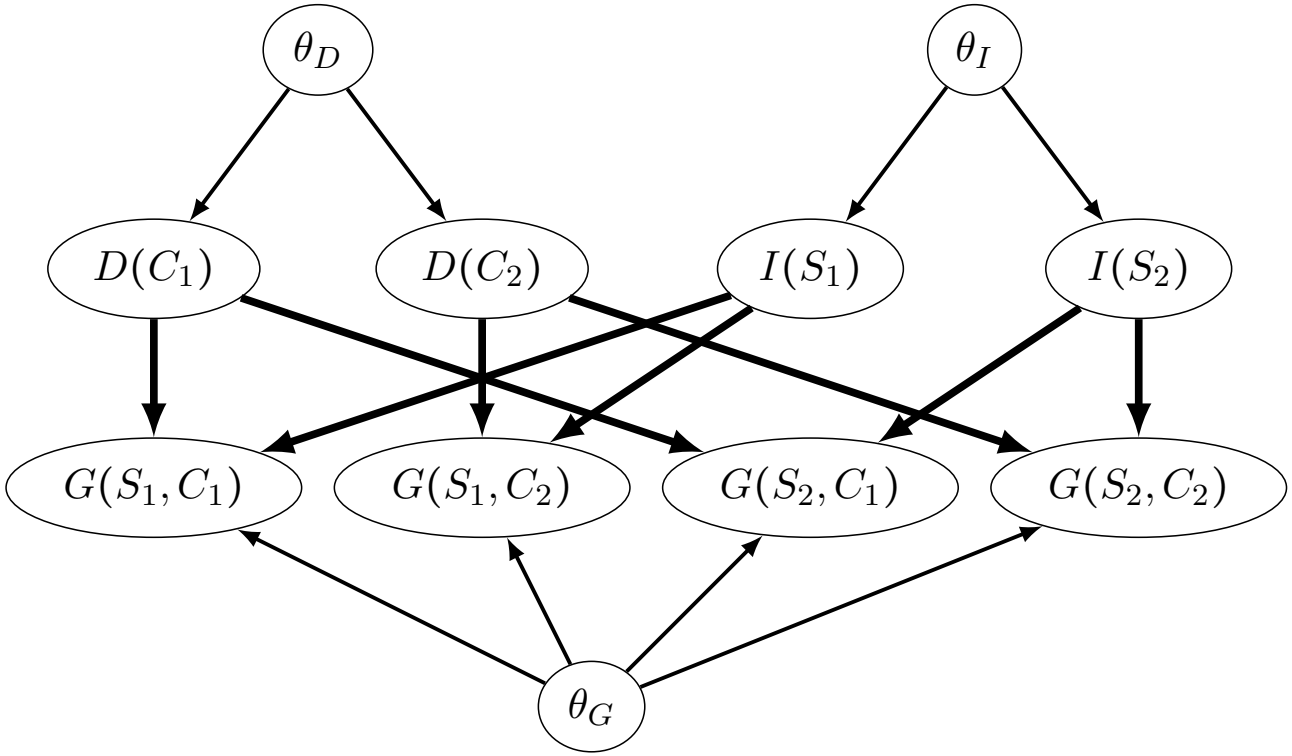


Plate models provide a language for encoding models with repeated structure.

Hierarchical Bayesian Networks using Plates

Complex DBNs can also be represented with plates