Introduction to Modeling Probability with Graphs

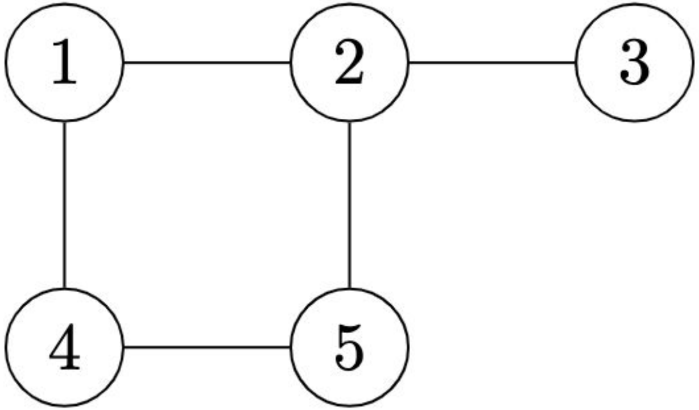
Douglas Allen

Overview

1. Brief Intro to Graphs
2. Modeling Probability Distributions with Graphs
3. Bayesian Networks and Markov Random Fields
4. Inference and Learning on Graphs
5. Areas of Further Learning

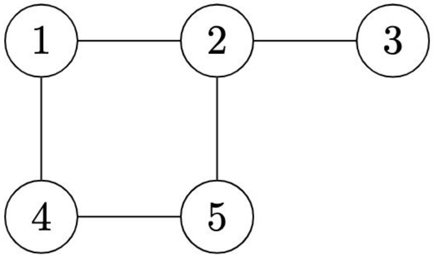
Intro to Graphs

A graph consists of a series of nodes and edges



Intro to Graphs

It can be represented as a drawing, a node-edge list, or an adjacency matrix



$$\equiv$$

$$G = (V, E)$$
$$V = \{1, 2, 3, 4, 5\}$$
$$E = \{(1,2), (1,4), (2,3), (2,5), (5,4)\}$$

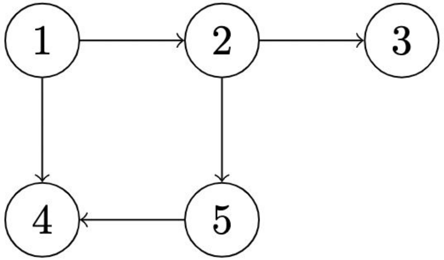
$$\equiv$$

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

This is a simple, undirected graph

Intro to Graphs

Graphs can be directed (and can contain self-edges or multi-edges, not covered here)



$$\equiv \quad \begin{aligned} G &= (V, E) \\ V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1,2), (1,4), (2,3), (2,5), (5,4)\} \end{aligned} \quad \equiv \quad A_G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

This is now a Directed, Acyclic Graph (DAG)

Intro to Graphs

Some other key components to graphs include:

- Connectedness
- Degree of a node
- Types of Graphs
- Subgraphs

Modeling Probability Distributions on Graphs

Modeling Probability Distributions on Graphs

Consider a joint probability distribution of five random variables:

$$P(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4, X_5 = x_5)$$

or more compactly: $p(x_1, x_2, x_3, x_4, x_5)$

With the chain rule of probability, we can break it down (factorize it) as five different terms:

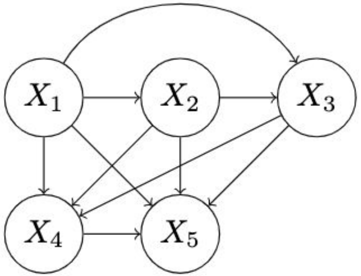
$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5|x_1, x_2, x_3, x_4)p(x_4|x_1, x_2, x_3)p(x_3|x_1, x_2)p(x_2|x_1)p(x_1)$$

(note this was an arbitrary decision, we could have factorized it in another way)

Modeling Probability Distributions on Graphs

Then we can represent this joint distribution as a graph

$$p(x_5|x_1, x_2, x_3, x_4)p(x_4|x_1, x_2, x_3)p(x_3|x_1, x_2)p(x_2|x_1)p(x_1)$$



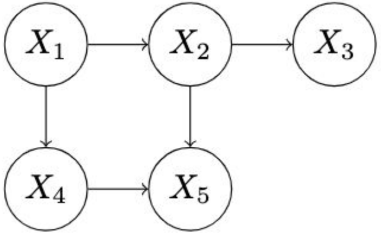
If we assume some variables are conditionally independent, we can get a cleaner graph that provides more insight into how data from this distribution was generated

$$x_3 \perp x_1|x_2$$

$$x_4 \perp \{x_2, x_3\}|x_1$$

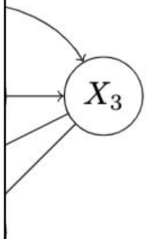
$$x_5 \perp \{x_1, x_3\}|x_2$$

$$p(x_5|x_2, x_4)p(x_4|x_1)p(x_3|x_2)p(x_2|x_1)p(x_1)$$



Modeling Probability Distributions on Graphs

Then we can represent this joint distributio



$$p(x_5|x_1, x_2, x_3, x_4)p(x_4|x_1, x_2, x_3)p(x_3|x_1, x_2)$$

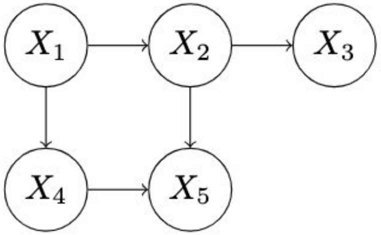
Here we typically refer to the variables that another 'depends' on as ancestor variables

For example: $x_{A(5)} = \{x_2, x_4\}$

If we assume some variables are condition leaner graph that provides more insight into how data from this distribution was generated

$$x_3 \perp x_1 | x_2$$
$$x_4 \perp \{x_2, x_3\} | x_1$$
$$x_5 \perp \{x_1, x_3\} | x_2$$

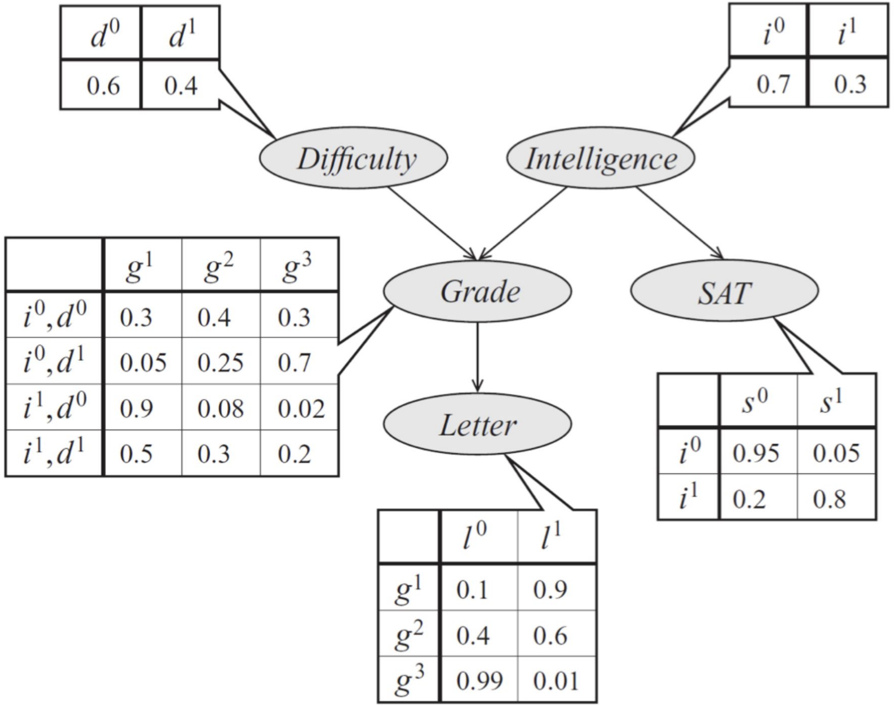
$$p(x_5|x_2, x_4)p(x_4|x_1)p(x_3|x_2)p(x_2|x_1)p(x_1)$$



Bayesian Networks and Markov Random Fields

Bayesian Network Example

Here is a real-world model for the performance of a student based on underlying attributes:



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

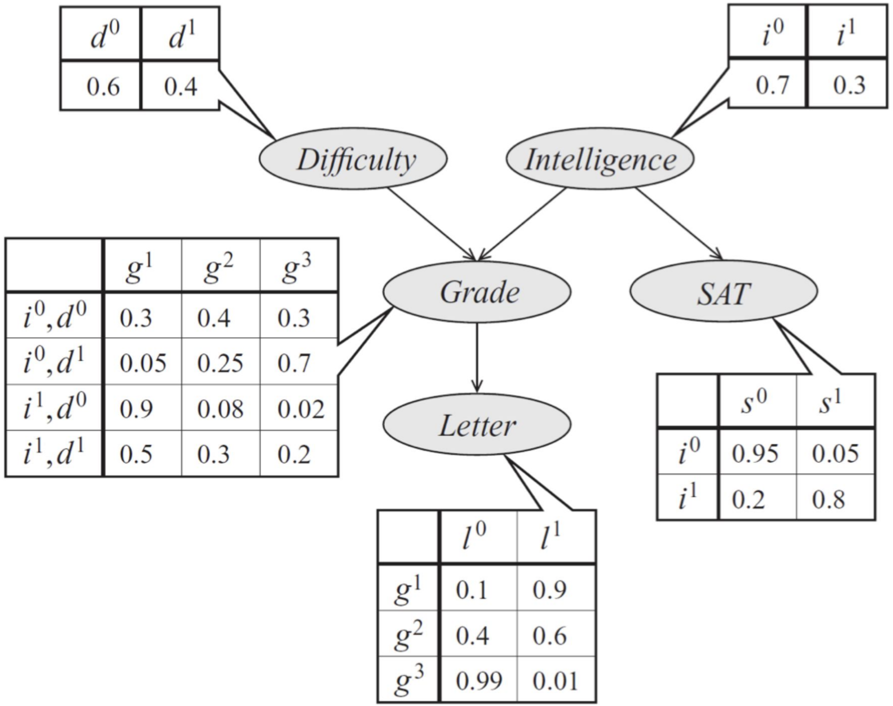
	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Bayesian Network Example

In this example, we have five variables of a student and their academic performance

Difficulty of a class, intelligence, grade, SAT score, and letter of recommendation sentiment

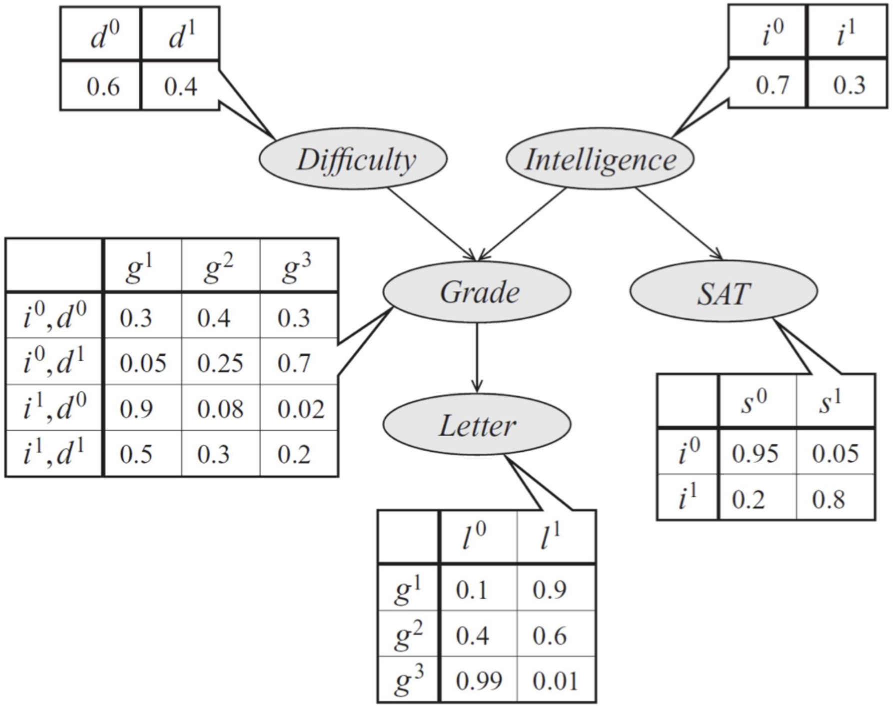
The graph provides a set of conditional independencies for these random variables



Bayesian Network Example

It also provides a coherent ‘story’ about how data is generated

This is useful when describing results to non-technical stakeholders



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Modeling Probability Distributions as Graphs

What if we don't want to assume or model anything about causality?

Imagine we are predicting voting habits among people, and we model them as a graph where people are nodes and edges are friendships

Each person can vote Yes on an issue $X_j = 1$, or no $X_j = 0$

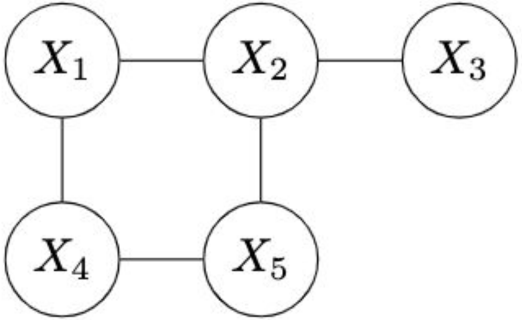
We can make a scoring function that maps to higher values if two nodes are connected (random variables are correlated):

$$\phi : E \to \mathbb{R}$$

$$\phi((X_i, X_j)) = \begin{cases} 10 \; X_i = X_j = 1 \\ 5 \; X_i = X_j = 0 \\ 1 \; X_i \neq X_j \end{cases}$$

Markov Random Field

We can use the previous construction to represent the probability distribution as an undirected graph



$$p(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} \prod_{(X_i, X_j) \in E} \phi(x_i, x_j)$$

$$Z = \sum_{X_i, i=1,\ldots 5} \prod_{(x_i, x_j) \in E} \phi(x_i, x_j)$$

Markov Random Field

In general, we define Markov Random Fields in terms of a graph's cliques

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)$$

$$Z = \sum_{x_1, \dots, x_n} \prod_{c \in C} \phi(x_c)$$

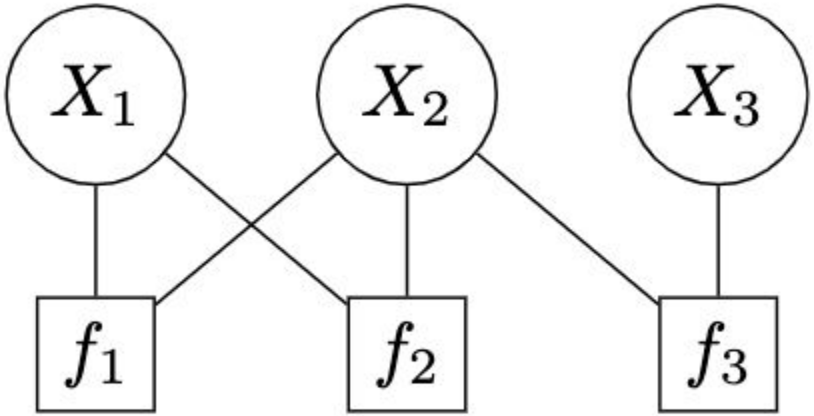
Where C is the set of cliques on the graph, and

$$\phi_c(x_c) \geq 0 \; \forall c \in C$$

Markov Random Field

A useful example of a Markov Random Field is a Factor Model

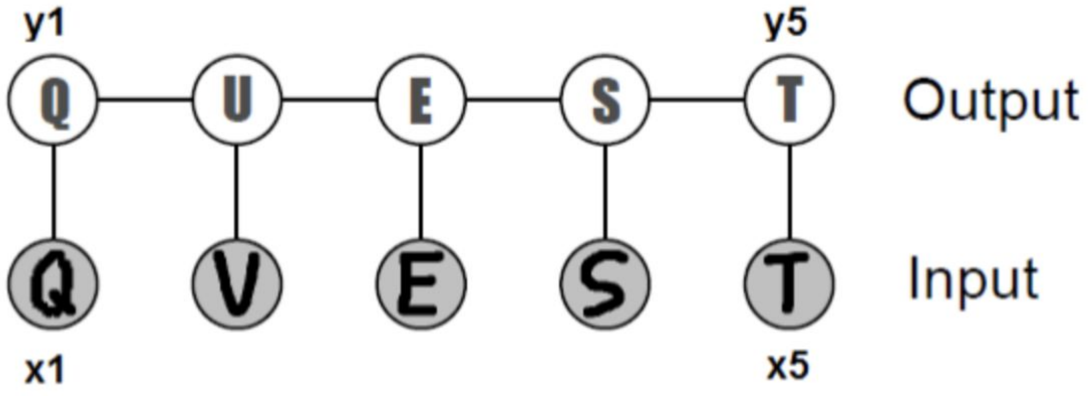
This is a bipartite graph with observed realizations of random variables, and factors conditioned only on those values



Markov Random Field

Another application of Markov Random Fields is conditional random fields, here used for text reading

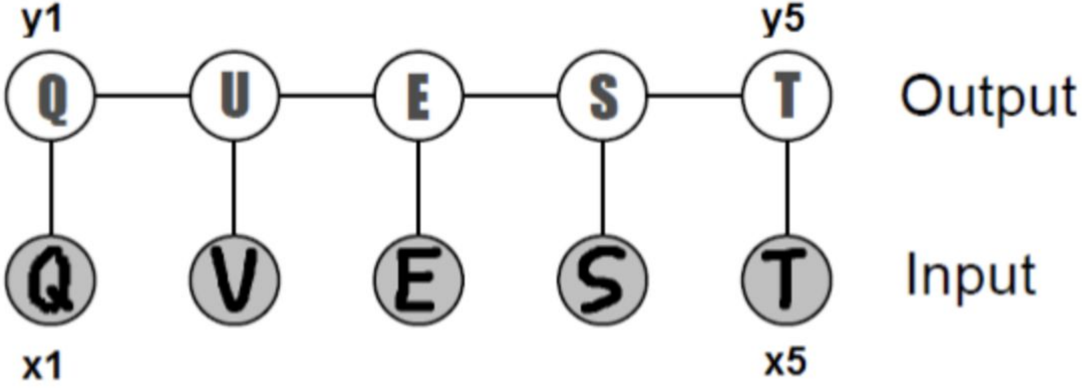
Rather than simply using a neural net to identify each character one at a time and concatenate the result, we can use conditions between characters to get a better estimate



Markov Random Field

In this example, an isolated classifier might guess the second letter is V

However when we combine score estimates with the fact that we're pretty certain the first letter is Q, we can get at much better score for the category of the second letter



So What?

Inference and Learning on Graphs

Inference (Calculating Probabilities) on Graphs

Inference and Learning on Graphs

Graphical models make it more computationally efficient to compute marginal probabilities

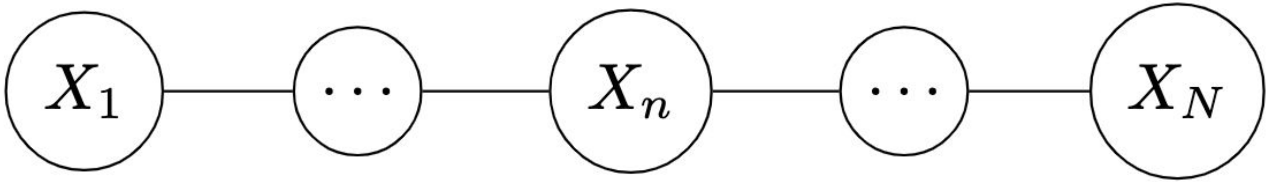
For example say we have N random variables that can each take K values, then there are $O(K^{N-1})$ computations in order to find the marginal PDF of one variable

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_n+1} \cdots \sum_{x_N} p(x_1, \ldots, x_N)$$

However if we tak e can reduce that dramatically

Inference on a Path

A simple example is if we know our model can be represented by a path graph representing a Markov Random Field,



Then every random variable is independent of the others unless they are adjacent on the graph

Inference on a Path

Then our expensive sum from earlier gets broken down:

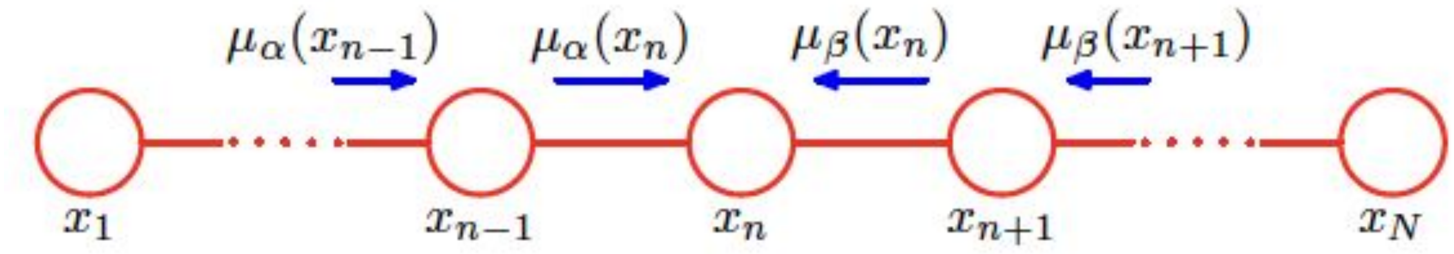
$$\left.\begin{aligned} p(x_n) = \frac{1}{Z}\Bigg(\sum_{x_{n-1}} \phi_{(n-1,n)}(x_{n-1}, x_n) \ldots \bigg(\sum_{x_2} \phi_{(2,3)}(x_2, x_3)\bigg(\sum_{x_1} \phi_{(1,2)}(x_1, x_2)\bigg)\bigg) \ldots \Bigg)\end{aligned}\right\} \leftarrow \mu_\alpha(x_n)$$

$$\left.\begin{aligned} \times\Bigg(\sum_{x_{n+1}} \phi_{(n,n+1)}(x_n, x_{n+1}) \ldots \bigg(\sum_{x_N} \phi_{(N-1,N)}(x_{N-1}, x_N)\bigg) \ldots \Bigg)\end{aligned}\right\} \leftarrow \mu_\beta(x_n)$$

It looks uglier, but it's actually $O((N-1)K^2)$

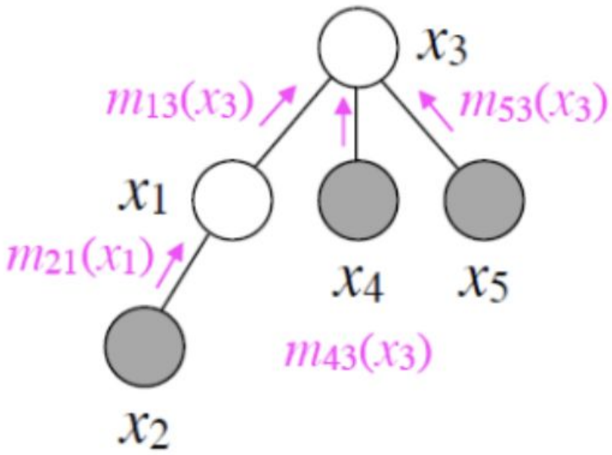
Inference on a Path

These two functions that multiply together are frequently thought of as ‘messages’ traveling from each end of the graph towards the desired node



Inference on a Tree

Now that we have an intuition for inference on a path, at a high level we can describe belief propagation



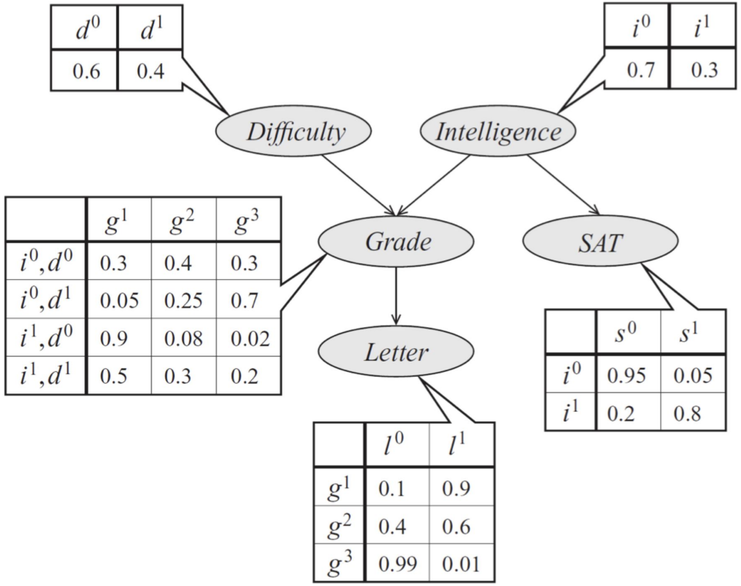
We can imagine multiple paths carrying messages up to node x_3 to compute its marginal probability

Learning the Parameters of a Graph

Learning on a Directed Network

We consider the simple case of MLE for a directed network

When we collect data, what will it look like? What are we trying to learn?



Each probability is a parameter $\theta_{x_i | x_{A(i)}}$

and the data look like:

	difficulty	intelligence	SAT	grade	letter
0	0.0	1.0	1.0	2.0	1.0
1	0.0	0.0	0.0	3.0	0.0
2	0.0	0.0	0.0	3.0	0.0
3	0.0	1.0	1.0	1.0	1.0
4	0.0	1.0	1.0	3.0	0.0
5	0.0	0.0	0.0	3.0	0.0
6	0.0	0.0	0.0	2.0	0.0
7	1.0	0.0	0.0	3.0	0.0
8	1.0	0.0	0.0	1.0	1.0
9	1.0	1.0	0.0	1.0	1.0

Learning on a Directed Network

As a quick note, another advantage of graphical models is how easy the make the process of creating synthetic data

To make this data, I simply had to write five multinoulli distributions

Three of these had parameters that were dependent on some of the previous variables, as determined by the graph

Then I generated a set of synthetic points easily, whereas trying to write code for one joint distribution would have been extremely difficult

	difficulty	intelligence	SAT	grade	letter
0	0.0	1.0	1.0	2.0	1.0
1	0.0	0.0	0.0	3.0	0.0
2	0.0	0.0	0.0	3.0	0.0
3	0.0	1.0	1.0	1.0	1.0
4	0.0	1.0	1.0	3.0	0.0
5	0.0	0.0	0.0	3.0	0.0
6	0.0	0.0	0.0	2.0	0.0
7	1.0	0.0	0.0	3.0	0.0
8	1.0	0.0	0.0	1.0	1.0
9	1.0	1.0	0.0	1.0	1.0

Learning on a Directed Network

I made a function gen_point which generates a datapoint based on the distribution,

```
def gen_point():
    if np.random.rand(1)<.6:
        d=0
    else:
        d=1

    if np.random.rand(1)<.7:
        i=0
        s0=.95
    else:
        i=1
        s0 = .2

    if np.random.rand(1)<s0:
        s=0
    else:
        s=1
```

```
    if d==0 and i==0:
        g1 = .3
        g2 = .7
    elif d==0 and i==1:
        g1 = .9
        g2 = .98
    elif d==1 and i==0:
        g1 = .05
        g2 = .3
    else:
        g1 = .5
        g2 = .8
```

```
    groll = np.random.rand(1)
    if groll < g1:
        g = 1
        l1 = .1
    elif groll < g2:
        g = 2
        l1 = .4
    else:
        g = 3
        l1 = .99

    if np.random.rand(1) < l1:
        l = 0
    else:
        l = 1

    point = np.array([d,i,s,g,l])
    point = point[None,:]

    return point
```

MLE on a Directed Network

Given a dataset: $\mathcal{D} = \{x^{(1)}, \ldots, x^{(m)}\}$ where $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$

We can calculate the likelihood:

$$L(\theta, \mathcal{D}) = \prod_{j=1}^{m} \prod_{i=1}^{n} \theta_{x_i^{(j)} | x_{A(i)}^{(j)}}$$

When we take the log and combine like terms we get

$$ll(\theta, \mathcal{D}) = \sum_{i=1}^{n} \sum_{x_{A(i)}} \sum_{x_i} \#(x_i, x_{A(i)}) \log \theta_{x_i^{(j)} | x_{A(i)}^{(j)}}$$

Maximizing gives us

$$\hat{\theta}_{x_i | x_{A(i)}} = \frac{\#(x_i, x_{A(i)})}{\#(x_{A(i)})}$$

MLE on a Directed Network

The code for this estimator is easy, and would probably be easier if I were better at coding

For example, to determine the grade parameters when the difficulty of the class is 0 and intelligence is 0, we write the following:

In [16]:
```
g1d0i0 = df[(df['grade']==1)&(df['difficulty']<1) & (df['intelligence']<1)].shape[0]/d0i0
g2d0i0 = df[(df['grade']==2)&(df['difficulty']<1) & (df['intelligence']<1)].shape[0]/d0i0
g3d0i0 = 1-(g1d0i0+g2d0i0)
print(g1d0i0,g2d0i0,g3d0i0)
```

```
0.30510846745976206 0.3949148588756706 0.29997667366456726
```

Here, d0i0 is the number of instances where difficulty is 0 and intelligence is 0

Recall true parameters .3, .4, and .3 - this simple implementation produces accurate results

Areas of Further Learning

There are many more topics of graphical models worth exploring:

- Sum-product algorithm
- Markov Chain Monte Carlo
- Learning Latent Variable Models
- Structure Learning

Sources

C. Bishop. *Pattern Recognition and Machine Learning*. 2006. ISBN: 978-0387-31073-2.

S. Ermon. *Stanford CS 288 Lecture Notes*. 2023. URL: `https://ermongroup.github.io/cs228-notes/`.

K. Murphy. *Machine Learning: A Probabilistic Perspective*. 2012. ISBN: 978-0-262-01802-9.

E Xing. *CMU CS 10-708 Lecture Notes*. 2014. URL: `http://www.cs.cmu.edu/~epxing/Class/10708-14/index.html`.