Learning

 Wilker Aziz

w.aziz@uva.nl

Fall 2025 (v3)

https://probabll.github.io

Outline and goals

Module 5 introduces *Learning Algorithms* for BNs and MNs (Chapters 17 and 20).

ILOs After this module the student

- can estimate parameters for a BN via MLE;
- can estimate parameters for an MN via (approximations to) MLE;

Textbook for this course: Koller and Friedman [1].

Overview of Module 1

HC6a: Parameter estimation for BNs.

LC6: Parameter estimation in code.

HC6b: Parameter estimation for MNs.

WC6: exercises.

Table of contents

Parameter Estimation

Prescribing a Categorical Distribution

Let X be a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$.

Prescribing a Categorical Distribution

Let X be a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$.

If we model with **tabular CPDs**, we can say that there is a 'table' (rather a 'row vector') $\boldsymbol{\theta} = (\theta_{x^1}, \ldots, \theta_{x^K})$ which stores the K **parameters** needed to prescribe the distribution of X:

- for any $x \in \mathbf{Val}(X)$, $0 \leq \theta_x \leq 1$
- $\sum_{x \in \mathbf{Val}(X)} \theta_x = 1$

Then, under this choice, $P(X = x) = \theta_x$.

Prescribing a Categorical Distribution

Let X be a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$.

If we model with **tabular CPDs**, we can say that there is a 'table' (rather a 'row vector') $\boldsymbol{\theta} = (\theta_{x^1}, \ldots, \theta_{x^K})$ which stores the K **parameters** needed to prescribe the distribution of X:

- for any $x \in \mathbf{Val}(X)$, $0 \leq \theta_x \leq 1$
- $\sum_{x \in \mathbf{Val}(X)} \theta_x = 1$

Then, under this choice, $P(X = x) = \theta_x$.

Suppose we have a dataset $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of M observations of realisations of X.

How can we use **data** to inform our choice of numerical values for the parameters $\boldsymbol{\theta}$?

Prescribing a Categorical Distribution

Let X be a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$.

If we model with **tabular CPDs**, we can say that there is a 'table' (rather a 'row vector') $\boldsymbol{\theta} = (\theta_{x^1}, \ldots, \theta_{x^K})$ which stores the K **parameters** needed to prescribe the distribution of X:

- for any $x \in \mathbf{Val}(X)$, $0 \leq \theta_x \leq 1$
- $\sum_{x \in \mathbf{Val}(X)} \theta_x = 1$

Then, under this choice, $P(X = x) = \theta_x$.

Suppose we have a dataset $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of M observations of realisations of X.

How can we use **data** to inform our choice of numerical values for the parameters $\boldsymbol{\theta}$? This is a **statistical inference** problem.

Frequentist Inference

First, characterise the model's likelihood function given the available data $\mathcal{D}$:

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} P(X = x[m]) = \prod_{m=1}^{M} \theta_{x[m]}$$

The likelihood function $L(\boldsymbol{\theta}; \mathcal{D})$ assigns 'worth' or 'utility' to a choice of parameter $\boldsymbol{\theta}$. This utility is defined to be the probability mass that our model assigns to $\mathcal{D}$ under the assumption that the data samples were drawn IID from our model using the current choice of $\boldsymbol{\theta}$.

Frequentist Inference

First, characterise the model's likelihood function given the available data $\mathcal{D}$:

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} P(X = x[m]) = \prod_{m=1}^{M} \theta_{x[m]}$$

The likelihood function $L(\boldsymbol{\theta}; \mathcal{D})$ assigns 'worth' or 'utility' to a choice of parameter $\boldsymbol{\theta}$. This utility is defined to be the probability mass that our model assigns to $\mathcal{D}$ under the assumption that the data samples were drawn IID from our model using the current choice of $\boldsymbol{\theta}$.

Second, pick the parameter value that yields maximum likelihood:

$$\boldsymbol{\theta}^{\star} = \underset{\boldsymbol{\theta} \in \Delta_{K-1}}{\operatorname{argmax}} \; L(\boldsymbol{\theta}; \mathcal{D}) = \underset{\boldsymbol{\theta} \in \Delta_{K-1}}{\operatorname{argmax}} \; \underbrace{\log L(\boldsymbol{\theta}; \mathcal{D})}_{\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})}$$

Frequentist Inference

First, characterise the model's likelihood function given the available data $\mathcal{D}$:

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} P(X = x[m]) = \prod_{m=1}^{M} \theta_{x[m]}$$

The likelihood function $L(\boldsymbol{\theta}; \mathcal{D})$ assigns 'worth' or 'utility' to a choice of parameter $\boldsymbol{\theta}$. This utility is defined to be the probability mass that our model assigns to $\mathcal{D}$ under the assumption that the data samples were drawn IID from our model using the current choice of $\boldsymbol{\theta}$.

Second, pick the parameter value that yields maximum likelihood:

$$\boldsymbol{\theta}^{\star} = \underset{\boldsymbol{\theta} \in \Delta_{K-1}}{\operatorname{argmax}} \; L(\boldsymbol{\theta}; \mathcal{D}) = \underset{\boldsymbol{\theta} \in \Delta_{K-1}}{\operatorname{argmax}} \; \underbrace{\log L(\boldsymbol{\theta}; \mathcal{D})}_{\mathcal{L}(\boldsymbol{\theta}; \mathcal{D})} = \sum_{m=1}^{M} \log \theta_{x[m]}$$

This is known as **maximum likelihood estimation (MLE).**

Bayesian Inference

Bayesian inference treats parameters as rvs on their own right.

It also uses the likelihood function, but in a very different way, as a means to update beliefs about parameters:

$$\underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{\theta})}_{\text{prior}} \underbrace{L(\boldsymbol{\theta};\mathcal{D})}_{\text{likelihood}}$$

Rather than looking for a parameter value judged to be 'right' (or 'optimum')—a point estimate—Bayesian inference attempts to estimate parameters that are *probable* in light of the available data and model assumptions as captured by the likelihood function.

Bayesian Inference

Bayesian inference treats parameters as rvs on their own right.

It also uses the likelihood function, but in a very different way, as a means to update beliefs about parameters:

$$\underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{\text{posterior}} \propto \underbrace{p(\boldsymbol{\theta})}_{\text{prior}} \underbrace{L(\boldsymbol{\theta};\mathcal{D})}_{\text{likelihood}}$$

Rather than looking for a parameter value judged to be 'right' (or 'optimum')—a point estimate—Bayesian inference attempts to estimate parameters that are *probable* in light of the available data and model assumptions as captured by the likelihood function.

In this course, we will focus on Frequentist inference (not because it's 'right' or to be preferred in general, simply because Bayesian inference requires a longer course).

MLE for the Parameters of a Categorical Distribution

X is a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$. We model its distribution in tabular form, that is, we introduce a parameter vector $\boldsymbol{\theta}_X = (\theta_{x^1}, \ldots, \theta_{x^K})$ such that $P(X = x) = \theta_x$.

We have a dataset $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of M observations of X.

Define the helper 'counting' function $M[o] = \sum_{m=1}^{M} [x[m] = o]$.

The Iverson bracket $[\alpha]$ is 1 if the logical predicate α is True and 0 otherwise. MLE for Categorical distribution: if you're curious I derived it here.

MLE for the Parameters of a Categorical Distribution

X is a categorical rv with outcomes in $\mathbf{Val}(X) = \{x^1, \ldots, x^K\}$. We model its distribution in tabular form, that is, we introduce a parameter vector $\boldsymbol{\theta}_X = (\theta_{x^1}, \ldots, \theta_{x^K})$ such that $P(X = x) = \theta_x$.

We have a dataset $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of M observations of X.

Define the helper 'counting' function $M[o] = \sum_{m=1}^{M} [x[m] = o]$.

The maximum likelihood estimate of θ_x for any $x \in \mathbf{Val}(X)$ is given by

$$\theta_x = \frac{M[x]}{\sum_{x' \in \mathbf{Val}(X)} M[x']} = \frac{M[x]}{M} \tag{1}$$

The Iverson bracket $[\alpha]$ is 1 if the logical predicate α is True and 0 otherwise. MLE for Categorical distribution: if you're curious I derived it here.

MLE for a Categorical Distribution – Example

The coherence C of a course may be high c^1 or low c^0. Let's do MLE for a tabular representation of $P(C)$.

Course	Votes
CS0001	$c^1, c^1, c^1, c^0, c^0, c^0, c^0, c^1, c^1, c^0$
CS0002	$c^1, c^1, c^1, c^0, c^0, c^0, c^1, c^1, c^1, c^1$

Observations from course evaluation surveys.

MLE for $P(C)$ using the data above: $\boldsymbol{\theta} = (\theta_{c^0}, \theta_{c^1}) = (\quad , \quad)$.

MLE for a Categorical Distribution – Example

The coherence C of a course may be high c^1 or low c^0. Let's do MLE for a tabular representation of $P(C)$.

Course	Votes
CS0001	$c^1, c^1, c^1, c^0, c^0, c^0, c^0, c^1, c^1, c^0$
CS0002	$c^1, c^1, c^1, c^0, c^0, c^0, c^1, c^1, c^1, c^1$

Observations from course evaluation surveys.

MLE for $P(C)$ using the data above: $\boldsymbol{\theta} = (\theta_{c^0}, \theta_{c^1}) = (8/20, 12/20)$.

MLE for the Parameters of a Tabular CPD

Now, we also have an rv Y taking on values in $\mathbf{Val}(Y) = \{y^1, \ldots, y^V\}$.

We are modelling $P(Y|X)$ in tabular representation. Hence, we introduce a collection of parameter vectors $\boldsymbol{\theta}_{Y|X} = (\boldsymbol{\theta}_{Y|x^1}, \ldots, \boldsymbol{\theta}_{Y|x^K})$.

- Each $\boldsymbol{\theta}_{Y|x}$ is a vector $(\theta_{y^1|x}, \ldots, \theta_{y^V|x})$ for some $x \in \mathbf{Val}(X)$
- such that $P(Y = y|X = x) = \theta_{y|x}$.

Now our observations are M pairs $\mathcal{D} = \{(x[1], y[1]), \ldots, (x[M], y[M])\}$.

Define a new 'counting' function $M[c, o] = \sum_{m=1}^{M} [x[m] = c][y[m] = o]$.

MLE for the Parameters of a Tabular CPD

Now, we also have an rv Y taking on values in $\mathbf{Val}(Y) = \{y^1, \ldots, y^V\}$.

We are modelling $P(Y|X)$ in tabular representation. Hence, we introduce a collection of parameter vectors $\boldsymbol{\theta}_{Y|X} = (\boldsymbol{\theta}_{Y|x^1}, \ldots, \boldsymbol{\theta}_{Y|x^K})$.

- Each $\boldsymbol{\theta}_{Y|x}$ is a vector $(\theta_{y^1|x}, \ldots, \theta_{y^V|x})$ for some $x \in \mathbf{Val}(X)$
- such that $P(Y = y|X = x) = \theta_{y|x}$.

Now our observations are M pairs $\mathcal{D} = \{(x[1], y[1]), \ldots, (x[M], y[M])\}$.

Define a new 'counting' function $M[c, o] = \sum_{m=1}^{M} [x[m] = c][y[m] = o]$.

The MLE of $\theta_{y|x}$ for any $x \in \mathbf{Val}(X)$ and $y \in \mathbf{Val}(Y)$ is given by

$$\theta_{y|x} = \frac{M[x, y]}{\sum_{y' \in \mathbf{Val}(Y)} M[x, y']} = \frac{M[x, y]}{M[x]} \tag{2}$$

MLE for a Categorical CPD – Example

The course may be difficult d^1 or easy d^0. Let's do MLE for a tabular representation of $P(D|C)$.

Course	Votes
CS0001	$(c^1, d^0), (c^1, d^0), (c^1, d^0), (c^0, d^1), (c^0, d^1)$ $(c^0, d^0), (c^0, d^1), (c^1, d^0), (c^1, d^0), (c^0, d^1)$
CS0002	$(c^1, d^1), (c^1, d^1), (c^1, d^1), (c^0, d^1), (c^0, d^1)$ $(c^0, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^0)$

Observations from course evaluation surveys.

MLE for $P(D|C)$ using the data above:

$\boldsymbol{\theta}_{D|c^0} = (\theta_{d^0|c^0}, \theta_{d^1|c^0}) = (\quad , \quad)$

$\boldsymbol{\theta}_{D|c^1} = (\theta_{d^0|c^1}, \theta_{d^1|c^1}) = (\quad , \quad)$

MLE for a Categorical CPD – Example

The course may be difficult d^1 or easy d^0. Let's do MLE for a tabular representation of $P(D|C)$.

Course	Votes
CS0001	$(c^1, d^0), (c^1, d^0), (c^1, d^0), (c^0, d^1), (c^0, d^1)$
	$(c^0, d^0), (c^0, d^1), (c^1, d^0), (c^1, d^0), (c^0, d^1)$
CS0002	$(c^1, d^1), (c^1, d^1), (c^1, d^1), (c^0, d^1), (c^0, d^1)$
	$(c^0, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^0)$

Observations from course evaluation surveys.

MLE for $P(D|C)$ using the data above:

$\boldsymbol{\theta}_{D|c^0} = (\theta_{d^0|c^0}, \theta_{d^1|c^0}) = (1/8, 7/8)$

$\boldsymbol{\theta}_{D|c^1} = (\theta_{d^0|c^1}, \theta_{d^1|c^1}) = (\quad, \quad)$

MLE for a Categorical CPD – Example

The course may be difficult d^1 or easy d^0. Let's do MLE for a tabular representation of $P(D|C)$.

Course	Votes
CS0001	$(c^1, d^0), (c^1, d^0), (c^1, d^0), (c^0, d^1), (c^0, d^1)$
	$(c^0, d^0), (c^0, d^1), (c^1, d^0), (c^1, d^0), (c^0, d^1)$
CS0002	$(c^1, d^1), (c^1, d^1), (c^1, d^1), (c^0, d^1), (c^0, d^1)$
	$(c^0, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^1), (c^1, d^0)$

Observations from course evaluation surveys.

MLE for $P(D|C)$ using the data above:

$$\boldsymbol{\theta}_{D|c^0} = (\theta_{d^0|c^0}, \theta_{d^1|c^0}) = (1/8, 7/8)$$

$$\boldsymbol{\theta}_{D|c^1} = (\theta_{d^0|c^1}, \theta_{d^1|c^1}) = (6/12, 6/12)$$

MLE for Bayesian Networks

MLE for Bayesian Networks

In a BN we have a CPD for each node X_i given the node's parents $\text{Pa}(X_i)$. For CPDs in tabular representation, this means we have a collection of parameters $\boldsymbol{\theta}_{X_i|\text{Pa}(X_i)}$ for each node X_i.

MLE for Bayesian Networks

In a BN we have a CPD for each node X_i given the node's parents $\text{Pa}(X_i)$. For CPDs in tabular representation, this means we have a collection of parameters $\boldsymbol{\theta}_{X_i|\text{Pa}(X_i)}$ for each node X_i.

The likelihood $L(\boldsymbol{\theta}; \mathcal{D})$ of a BN is $L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} P(\boldsymbol{X} = \boldsymbol{x}[m])$

$$= \prod_{m=1}^{M} \prod_{i} P(X_i = x[m]_i | \text{Pa}(X_i) = \text{pa}(x[m]_i))$$

MLE for Bayesian Networks

In a BN we have a CPD for each node X_i given the node's parents $\text{Pa}(X_i)$. For CPDs in tabular representation, this means we have a collection of parameters $\boldsymbol{\theta}_{X_i|\text{Pa}(X_i)}$ for each node X_i.

The likelihood $L(\boldsymbol{\theta}; \mathcal{D})$ of a BN is $L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} P(\boldsymbol{X} = \boldsymbol{x}[m])$

$$= \prod_{m=1}^{M} \prod_{i} P(X_i = x[m]_i | \text{Pa}(X_i) = \text{pa}(x[m]_i))$$

$$= \prod_{i} \underbrace{\prod_{m=1}^{M} P(X_i = x[m]_i | \text{Pa}(X_i) = \text{pa}(x[m]_i))}_{L(\boldsymbol{\theta}_{X_i|\text{Pa}(X_i)}; \mathcal{D})}$$

decomposes in terms of **local likelihood functions**, one per X_i, where

$$L(\boldsymbol{\theta}_{X_i|\text{Pa}(X_i)}; \mathcal{D}) = \prod_{m=1}^{M} \theta_{x[m]_i|\text{pa}(x[m]_i)}$$

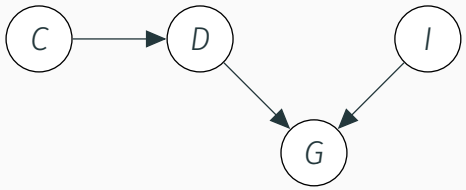
MLE for Bayesian Networks - Algorithm

Because the BN likelihood decomposes, so long as we have complete observations (that is, we learn by observing joint assignments of *all* rvs), we can solve independent MLE problems, one per rv.

For each rv X_i, the MLE for the parameters of the tabular CPD $P(X_i|\text{Pa}(X_i))$ is

- $\theta_{o|\boldsymbol{u}} = \frac{M[\boldsymbol{u},o]}{M[\boldsymbol{u}]}$ for any $o \in \textbf{Val}(X_i)$ and any $\boldsymbol{u} \in \textbf{Val}(\text{Pa}(X_i))$

MLE for BNs - Example



Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

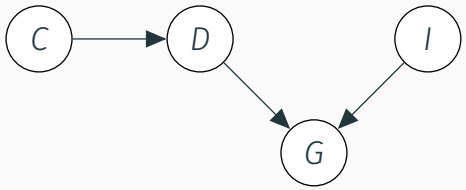
Observations from CS0001.

	θ_{i^0}	θ_{i^1}

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0			
d^0, i^1			
d^1, i^0			
d^1, i^1			

MLE for $P(I)$ and $P(G|D, I)$

MLE for BNs - Example



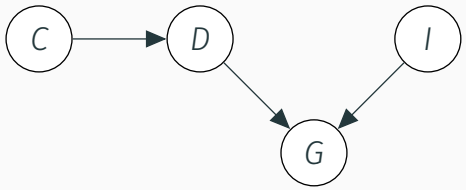
Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

Observations from CS0001.

	θ_{i^0}	θ_{i^1}
	$3/10$	$7/10$

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0			
d^0, i^1			
d^1, i^0			
d^1, i^1			

MLE for $P(I)$ and $P(G|D, I)$



Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

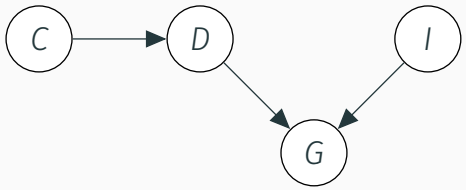
Observations from CS0001.

θ_{i^0}	θ_{i^1}
$3/10$	$7/10$

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0	$1/2$	$0/2$	$1/2$
d^0, i^1			
d^1, i^0			
d^1, i^1			

MLE for $P(I)$ and $P(G|D, I)$

MLE for BNs - Example



Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

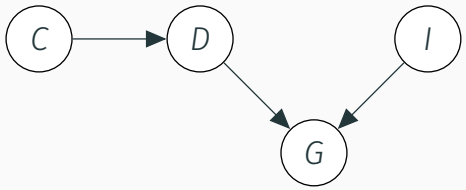
Observations from CS0001.

	θ_{i^0}	θ_{i^1}
	$3/10$	$7/10$

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0	$1/2$	$0/2$	$1/2$
d^0, i^1	$2/4$	$2/4$	$0/3$
d^1, i^0			
d^1, i^1			

MLE for $P(I)$ and $P(G|D, I)$

MLE for BNs - Example



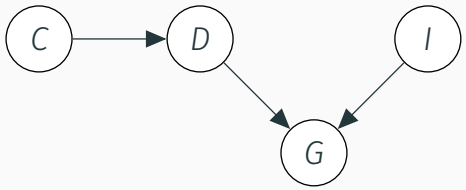
Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

Observations from CS0001.

	θ_{i^0}	θ_{i^1}
	$3/10$	$7/10$

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0	$1/2$	$0/2$	$1/2$
d^0, i^1	$2/4$	$2/4$	$0/3$
d^1, i^0	$1/1$	$0/1$	$0/1$
d^1, i^1			

MLE for $P(I)$ and $P(G|D, I)$



Student	Record
s1000	(c^0, d^1, i^1, g^1)
s1001	(c^0, d^0, i^1, g^1)
s1002	(c^0, d^1, i^0, g^1)
s1003	(c^1, d^0, i^1, g^2)
s1004	(c^1, d^0, i^1, g^2)
s1005	(c^1, d^0, i^0, g^1)
s1006	(c^1, d^0, i^0, g^3)
s1007	(c^0, d^1, i^1, g^3)
s1008	(c^0, d^1, i^1, g^2)
s1009	(c^1, d^0, i^1, g^1)

Observations from CS0001.

	θ_{i^0}	θ_{i^1}
	$3/10$	$7/10$

pa	$\theta_{g^1\|d,i}$	$\theta_{g^2\|d,i}$	$\theta_{g^3\|d,i}$
d^0, i^0	$1/2$	$0/2$	$1/2$
d^0, i^1	$2/4$	$2/4$	$0/3$
d^1, i^0	$1/1$	$0/1$	$0/1$
d^1, i^1	$1/3$	$1/3$	$1/3$

MLE for $P(I)$ and $P(G|D, I)$

Data Sparsity

Take the example from the previous slide: $\boldsymbol{\theta}_{G|d^1,i^0}$ whose MLE is $(1/1, 0/1, 0/1)$. Clearly, this MLE must not be very robust, after all, it is based on a single observation of $(D = d^1, I = i^0)$.

Data Sparsity

Take the example from the previous slide: $\boldsymbol{\theta}_{G|d^1,i^0}$ whose MLE is $(1/1, 0/1, 0/1)$. Clearly, this MLE must not be very robust, after all, it is based on a single observation of $(D = d^1, I = i^0)$.

Frequentist estimation is very 'data-hungry', it suffers from 'data sparsity': we need large sample sizes in order to observe enough occurrences of all possible outcomes.

In our tabular CPDs, parameters are independent of one another (the probability you assign to g^1 has nothing to do with the probability you assign to g^2 in the same given context, or in different but similar contexts, except that they sum to 1).

Data Sparsity

Take the example from the previous slide: $\boldsymbol{\theta}_{G|d^1,i^0}$ whose MLE is $(1/1, 0/1, 0/1)$. Clearly, this MLE must not be very robust, after all, it is based on a single observation of $(D = d^1, I = i^0)$.

Frequentist estimation is very 'data-hungry', it suffers from 'data sparsity': we need large sample sizes in order to observe enough occurrences of all possible outcomes.

In our tabular CPDs, parameters are independent of one another (the probability you assign to g^1 has nothing to do with the probability you assign to g^2 in the same given context, or in different but similar contexts, except that they sum to 1).

Solutions to this take different forms: Bayesian estimation represents uncertainty around parameters, Frequentist estimation use 'regularisers' and/or increase parameter sharing.

Regularised MLE and MAP Estimation for Tabular CPDs

A 'regulariser' is a pressure to deviate from the MLE objective in some systematic way. It is common to perform **regularised MLE** instead of (pure) MLE:

$$\boldsymbol{\theta}^{\star} = \operatorname*{argmax}_{\boldsymbol{\theta}} \; \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) - \mathcal{R}(\boldsymbol{\theta}) \tag{3}$$

where $\mathcal{R}(\boldsymbol{\theta})$ is some form of 'penalty' on certain parameters values (for example, those that are too large or too sparse).

Some choices of $\mathcal{R}(\boldsymbol{\theta})$ can be regarded as a form of *prior* about how parameter values would distributed if they were given random treatment. Such objectives are often called maximum-a-posteriori (MAP) estimation, for they coincide with the argmax of the Bayesian posterior $p(\boldsymbol{\theta}|\mathcal{D})$.

Watch Out! MAP Inference $\neq$ MAP Estimation

Don't confuse MAP inference in PGMs (that is, max-product inference), which concerns a max/argmax query about the rvs of a model, to MAP estimation in frequentist statistics (a regularised likelihood objective), which concerns parameter estimation.

MAP Inference in PGMs:

$$\boldsymbol{x}^\star = \operatorname*{argmax}_{\boldsymbol{x} \in \mathrm{Val}(\boldsymbol{X})} P(\boldsymbol{X} = \boldsymbol{x})$$

$\boldsymbol{x}^\star$ is an assignment of the rvs that are jointly distributed under $P(\boldsymbol{X})$.

MAP Estimation in Frequentist Statistics:

$$\boldsymbol{\theta}^\star = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) - \mathcal{R}(\boldsymbol{\theta})$$

$\boldsymbol{\theta}^\star$ is a collection of parameters that give numerical values for the cells of the tabular CPDs that parameterise $P(\boldsymbol{X})$.

Laplace Smoothed MLE for Tabular Categorical CPDs

A 'patch' for situations where we don't have enough data to estimate our tabular CPDs is to 'smooth' the MLE by a 'pseudo-count' $\alpha > 0$: a count that we pretend all context-outcome pairs start from before we even gather observations.

The **Laplace smoothed** MLE of $\theta_{X|\boldsymbol{u}}$ for any $x \in \mathsf{Val}(X)$ and $\boldsymbol{u} \in \mathsf{Val}(\mathrm{Pa}(X))$ is given by

$$\theta_{x|\boldsymbol{u}} = \frac{M[\boldsymbol{u}, x] + \alpha}{\sum_{x' \in \mathsf{Val}(X)} (M[\boldsymbol{u}, x'] + \alpha)} = \frac{M[\boldsymbol{u}, x] + \alpha}{\alpha |\,\mathsf{Val}(X)| + M[\boldsymbol{u}]} \tag{4}$$

In some more advanced versions, α can be specified per context ($\boldsymbol{u}$).

Out of curiosity: Laplace smoothing (or 'add-α smoothing') corresponds to MAP estimation using a Dirichlet prior $\theta_{X|\boldsymbol{u}} \sim \mathsf{Dir}(\alpha)$.

Laplace (add-α) Smoothing - Example

With add-0.1 smoothing, the example of $\boldsymbol{\theta}_{G|d^1,i^0}$ becomes $(^{1.1}/_{1.3}, ^{0.1}/_{1.3}, ^{0.1}/_{1.3})$.

Note how 0.1 is added to all 3 outcomes of G and hence affects the denominator in triple dose.

Laplace (add-α) Smoothing - Example

With add-0.1 smoothing, the example of $\boldsymbol{\theta}_{G|d^1,i^0}$ becomes $(^{1.1}/_{1.3}, ^{0.1}/_{1.3}, ^{0.1}/_{1.3})$.

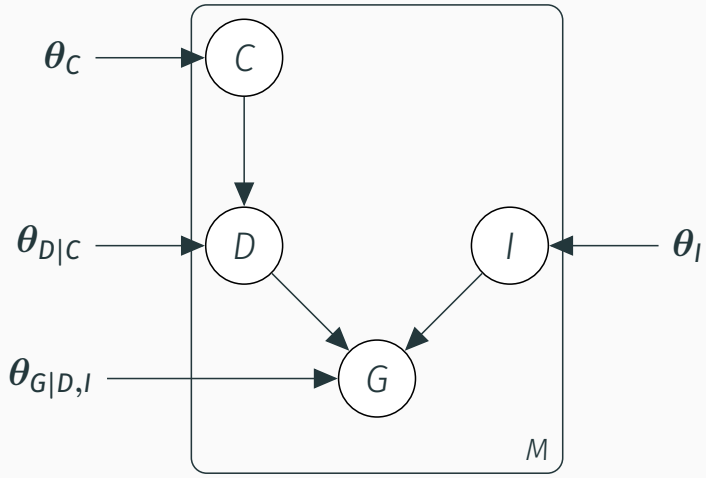
Note how 0.1 is added to all 3 outcomes of G and hence affects the denominator in triple dose.

This really is just a 'patch' to avoid 0s. Better strategies cpme from clever forms of parameter sharing. The first of which, we cover next.

Parameter Sharing with Template Models

It is very common to think of a PGM as a *template* to be instantiated given certain meta-data.

For example, we can imagine that our simplified student BN can be used to model M students who take a certain course:

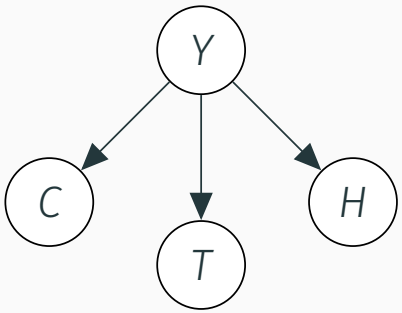


- Students are independent of one another.
- There are 4 tabular CPDs that are reused across all students.

Think of the plate as a 'for loop', the variables inside are 'instantiated' for each 'data record' out of M such data records. The assignments in one iteration are independent of the assignments in another. But all these iterations benefit from *shared* CPDs and their parameters (shown outside the plate).

Suppose we have 3 'predictors' C, T, H for a condition Y.

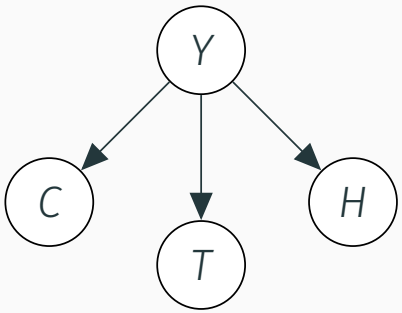
A condition might be something like COVID19 y^1 (True) or y^0 (False) and predictors might be things like: cough c^1 or c^0, high temperature t^1 or t^0, headache h^1 or h^0).



This is known as a **naive Bayes** model, commonly used for **classification** via $\textbf{argmax}_y \; P(Y = y | C = c, T = t, H = h)$ for some given assignment $(C = c, T = t, H = h)$ of the symptoms.

Suppose we have 3 'predictors' C, T, H for a condition Y.

A condition might be something like COVID19 y^1 (True) or y^0 (False) and predictors might be things like: cough c^1 or c^0, high temperature t^1 or t^0, headache h^1 or h^0).

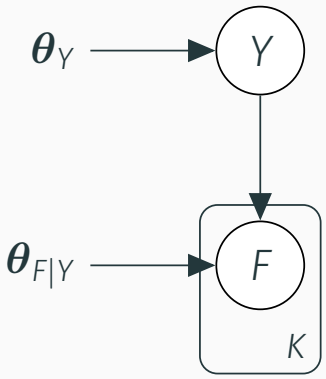


This is known as a **naive Bayes** model, commonly used for **classification** via $\textbf{argmax}_y \; P(Y = y|C = c, T = t, H = h)$ for some given assignment $(C = c, T = t, H = h)$ of the symptoms.

In a naive Bayes model, we assume that the distribution of $C|Y = y$ is the same as the distribution of $T|Y = y$ and of $H|Y = y$.

Instead of 3 CPDs $P(C|Y)$, $P(T|Y)$ and $P(H|Y)$, we have one CPD $P(F|Y)$ for 'feature value' given 'class' and we take c, t, h to be 3 feature values drawn from that CPD.

With K binary 'predictors' $X_1, \ldots, X_K$ (e.g., Cough, Temperature, Headache, loss of sense of Smell) for a condition Y. The NB model is a template BN.



Tabular CPDs

- $P(Y = y) = \theta_y$
- $P(F = f|Y = y) = \theta_{f|y}$

Compute the MLE parameters using the observations in the table.

Y	C	T	H	S
y^0	c^1	t^1	h^1	s^1
y^0	c^0	t^1	h^1	s^0
y^1	c^0	t^1	h^1	s^0
y^1	c^1	t^1	h^1	s^1
y^1	c^1	t^1	h^1	s^1
y^0	c^1	t^0	h^1	s^1
y^0	c^1	t^1	h^0	s^1
y^0	c^1	t^1	h^1	s^1
y^0	c^1	t^1	h^1	s^0
y^0	c^1	t^1	h^1	s^0
y^0	c^0	t^0	h^0	s^0
y^0	c^0	t^1	h^1	s^1
y^0	c^0	t^0	h^1	s^0
y^1	c^1	t^0	h^0	s^1
y^0	c^1	t^0	h^0	s^0
y^0	c^1	t^1	h^1	s^0
y^0	c^0	t^1	h^1	s^0
y^0	c^1	t^0	h^0	s^0
y^0	c^0	t^0	h^0	s^0
y^0	c^1	t^0	h^0	s^0

Solution

$\theta_Y = (64/80, 16/80)$

For $\boldsymbol{\theta}_{F|Y}$, see below:

Y	$\theta_{c^0\|y}$	$\theta_{c^0\|y}$	$\theta_{h^0\|y}$	$\theta_{h^1\|y}$	$\theta_{s^0\|y}$	$\theta_{s^1\|y}$	$\theta_{t^0\|y}$	$\theta_{t^1\|y}$
y^0	$6/64$	$10/64$	$6/64$	$10/64$	$11/64$	$5/64$	$7/64$	$9/64$
y^1	$1/16$	$3/16$	$1/16$	$3/16$	$1/16$	$3/16$	$1/16$	$3/16$

Because we treat the different symptoms as draws from the same CPD, we have 80 paired data points of the kind (Y, F). It is *as if* we had more data to estimate one CPD $P(F|Y)$ than we would have to estimate 4 CPDs $P(C|Y)$, $P(H|Y)$, $P(S|Y)$, $P(T|Y)$ instead.

Evaluating the MLE Solution

'Intrinsically'

- Assess $L(\boldsymbol{\theta}; \mathcal{H})$ for a dataset 'heldout' from training;

'Extrinsically' (task-driven)

- use the model in a predictive task (e.g., classification) and measure its task performance for some heldout dataset.

Summary

Learning BNs is relatively straightforward because the hierarchical structure of CPDs leads to a decomposed likelihood function.

With complete observations, the MLE of tabular CPDs is simply the frequency of the rv's outcome relative to the rv's parents' outcomes.

MLE suffers from data sparsity and mitigation strategies involve smoothing and/or parameter sharing.

In a course like NNs or DL you will learn how to design strategies for parameterising CPDs that enjoy a great deal of parameter sharing. (The combination of DL and PGMs is an exciting topic in advanced ML/DL).

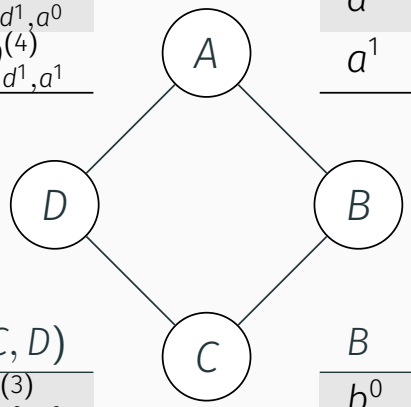
MLE for Markov Networks

Factor Values as Parameters

We begin by simply regarding the factor values as parameters. In the Misconception example,

D	A	$\phi_4(D, A)$
d^0	a^0	$\theta^{(4)}_{d^0,a^0}$
d^0	a^1	$\theta^{(4)}_{d^0,a^1}$
d^1	a^0	$\theta^{(4)}_{d^1,a^0}$
d^1	a^1	$\theta^{(4)}_{d^1,a^1}$

A	B	$\phi_1(A, B)$
a^0	b^0	$\theta^{(1)}_{a^0,b^0}$
a^0	b^1	$\theta^{(1)}_{a^0,b^1}$
a^1	b^0	$\theta^{(1)}_{a^1,b^0}$
a^1	b^1	$\theta^{(1)}_{a^1,b^1}$



C	D	$\phi_3(C, D)$
c^0	d^0	$\theta^{(3)}_{c^0,d^0}$
c^0	d^1	$\theta^{(3)}_{c^0,d^1}$
c^1	d^0	$\theta^{(3)}_{c^1,d^0}$
c^1	d^1	$\theta^{(3)}_{c^1,d^1}$

B	C	$\phi_2(B, C)$
b^0	c^0	$\theta^{(2)}_{b^0,c^0}$
b^0	c^1	$\theta^{(2)}_{b^0,c^1}$
b^1	c^0	$\theta^{(2)}_{b^1,c^0}$
b^1	c^1	$\theta^{(2)}_{b^1,c^1}$

We have 1 parameter vector per table $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \boldsymbol{\theta}^{(4)})$, each $\boldsymbol{\theta}^{(i)}$ is 4-dimensional since $\mathbf{Val}(\text{Scope}[\phi_i])$ has cardinality 4.

Partition Function

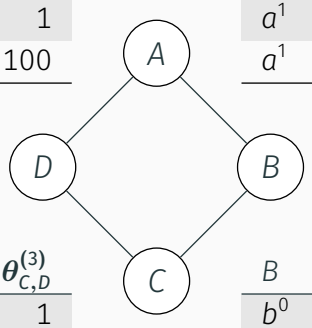
An MN over rvs $\boldsymbol{X}$ parameterised by a collection of factors Φ represents a distribution $P_\Phi(\boldsymbol{X})$.

When we consider the factor values as parameters $\boldsymbol{\theta}$, the unnormalised measure $\tilde{P}_\Phi(\boldsymbol{X};\boldsymbol{\theta})$ and the *normaliser* $Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{X}} \tilde{P}_\Phi(\boldsymbol{X};\boldsymbol{\theta})$ become functions of $\boldsymbol{\theta}$.

Example:

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



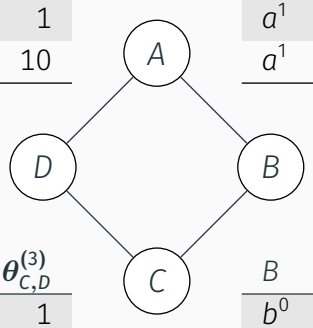
C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

$$P(A = a^0, B = b^0, C = c^0, D = d^0) = \frac{30 \times 100 \times 1 \times 100}{7201840}$$

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	10
d^0	a^1	1
d^1	a^0	1
d^1	a^1	10

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	3
a^0	b^1	5
a^1	b^0	1
a^1	b^1	1



C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	10
c^1	d^0	10
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	10
b^0	c^1	1
b^1	c^0	1
b^1	c^1	10

$$P(A = a^0, B = b^0, C = c^0, D = d^0) = \frac{3 \times 10 \times 1 \times 10}{7384}$$

Likelihood Function

Given observations $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, we can express the MN's likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} \frac{\tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{5}$$

Likelihood Function

Given observations $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, we can express the MN's likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} \frac{\tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{5}$$

whose logarithm is

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) =$$

Likelihood Function

Given observations $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, we can express the MN's likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} \frac{\tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{5}$$

whose logarithm is

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \sum_{m=1}^{M} \log \tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta})$$

Likelihood Function

Given observations $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, we can express the MN's likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} \frac{\tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{5}$$

whose logarithm is

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \sum_{m=1}^{M} \log \tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta})$$

The MN factorises, hence we can rewrite

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M}$$

Likelihood Function

Given observations $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, we can express the MN's likelihood function

$$L(\boldsymbol{\theta}; \mathcal{D}) = \prod_{m=1}^{M} \frac{\tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \tag{5}$$

whose logarithm is

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \sum_{m=1}^{M} \log \tilde{P}_{\Phi}(\boldsymbol{X} = \boldsymbol{x}[m]; \boldsymbol{\theta}) - \log Z(\boldsymbol{\theta})$$

The MN factorises, hence we can rewrite

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \Phi} \log \theta^{(i)}_{\mathbf{d}_i[m]}$$

where $\mathbf{d}_i[m]$ is the assignment to the rvs in the scope of factor i based on the mth observation $\boldsymbol{x}[m]$.

MLE

We look for the parameter value that maximises the log-likelihood function:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta} \in \mathbb{R}^D_{\geq 0}}{\operatorname{argmax}} \; -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}_i[m]}$$

Note that we optimise over the space of non-negative parameters (and we may even choose to focus on strictly positive parameters).

MLE

We look for the parameter value that maximises the log-likelihood function:

$$\boldsymbol{\theta}^{\star} = \underset{\boldsymbol{\theta} \in \mathbb{R}^{D}_{\geq 0}}{\operatorname{argmax}} \; -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}_i[m]}$$

Note that we optimise over the space of non-negative parameters (and we may even choose to focus on strictly positive parameters).

Challenge: the normaliser is a function of $\boldsymbol{\theta}$. Sum-product VE can help when the MN is sparsely connected.

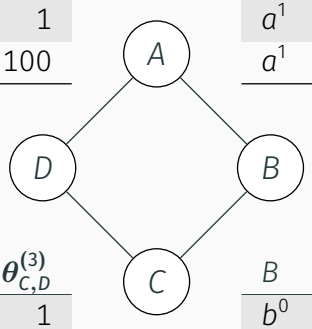
Unlike the case for BNs, there is no simple expression to obtain the global optimum.

Log-Likelihood - Example 1/2

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}_i[m]}$$

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

$Z = 7201840$

Log-likelihood:

$=$

Data:

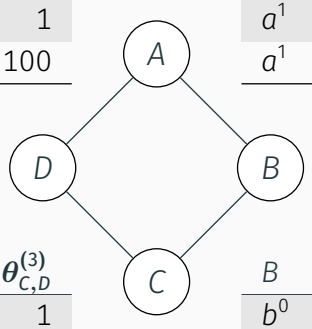
- a^0, b^0, c^1, d^1
- a^1, b^1, c^0, d^0
- a^0, b^1, c^1, d^0

Log-Likelihood - Example 1/2

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}_i[m]}$$

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

Data:

- a^0, b^0, c^1, d^1
- a^1, b^1, c^0, d^0
- a^0, b^1, c^1, d^0

$Z = 7201840$

Log-likelihood:

$$\begin{aligned}
&= -3 \times \log 7201840 + \\
&\quad + \log 30 + \log 1 + \log 1 + \log 1 \\
&\quad + \log 10 + \log 1 + \log 1 + \log 1 \\
&\quad + \log 5 + \log 100 + \log 100 + \log 100 \\
&= -26.2408
\end{aligned}$$

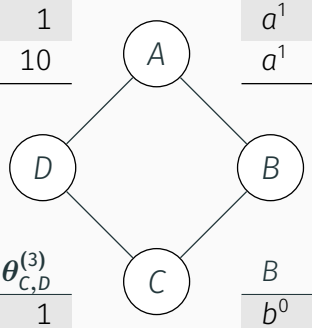
Log-Likelihood - Example 2/2

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}[m]_i}$$

$$Z = 7384$$

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	10
d^0	a^1	1
d^1	a^0	1
d^1	a^1	10

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	3
a^0	b^1	5
a^1	b^0	1
a^1	b^1	1



C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	10
c^1	d^0	10
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	10
b^0	c^1	1
b^1	c^0	1
b^1	c^1	10

Log-likelihood:

$=$

Data:

- a^0, b^0, c^1, d^1
- a^1, b^1, c^0, d^0
- a^0, b^1, c^1, d^0

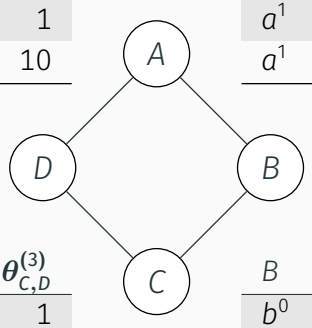
Log-Likelihood - Example 2/2

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = -M \log Z(\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{\phi_i \in \boldsymbol{\Phi}} \log \theta^{(i)}_{\mathbf{d}[m]_i}$$

$Z = 7384$

D	A	$\boldsymbol{\theta}^{(4)}_{D,A}$
d^0	a^0	10
d^0	a^1	1
d^1	a^0	1
d^1	a^1	10

A	B	$\boldsymbol{\theta}^{(1)}_{A,B}$
a^0	b^0	3
a^0	b^1	5
a^1	b^0	1
a^1	b^1	1



C	D	$\boldsymbol{\theta}^{(3)}_{C,D}$
c^0	d^0	1
c^0	d^1	10
c^1	d^0	10
c^1	d^1	1

B	C	$\boldsymbol{\theta}^{(2)}_{B,C}$
b^0	c^0	10
b^0	c^1	1
b^1	c^0	1
b^1	c^1	10

Data:

- a^0, b^0, c^1, d^1
- a^1, b^1, c^0, d^0
- a^0, b^1, c^1, d^0

Log-likelihood:

$$\begin{aligned}
&= -3 \times \log 7384 + \\
&\quad + \log 3 + \log 1 + \log 1 + \log 1 \\
&\quad + \log 1 + \log 1 + \log 1 + \log 1 \\
&\quad + \log 5 + \log 10 + \log 10 + \log 10 \\
&= -17.1054
\end{aligned}$$

In terms of log-likelihood, this version is preferred to the other.

Search

How can we search for the optimum?

Intuition: try a large number of cases (like we just did), assess the log-likelihood each time and pick the best.

But how many possibilities are there?

Search

How can we search for the optimum?

Intuition: try a large number of cases (like we just did), assess the log-likelihood each time and pick the best.

But how many possibilities are there? Uncountably many! Parameter values are *continuous* numbers.

Search

How can we search for the optimum?

Intuition: try a large number of cases (like we just did), assess the log-likelihood each time and pick the best.

But how many possibilities are there? Uncountably many! Parameter values are *continuous* numbers.

If we were to choose a grid of N values (for example, for each real-valued parameter try the number from -10 to 10 in increments of 0.1), with D parameters we need to try N^D parameter vectors.

What should we do?

Search

How can we search for the optimum?

Intuition: try a large number of cases (like we just did), assess the log-likelihood each time and pick the best.

But how many possibilities are there? Uncountably many! Parameter values are *continuous* numbers.

If we were to choose a grid of N values (for example, for each real-valued parameter try the number from -10 to 10 in increments of 0.1), with D parameters we need to try N^D parameter vectors.

What should we do? **Gradient-based optimisation**.

Gradient-Based Optimisation

Iterating

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma_t \nabla_{\boldsymbol{\theta}^{(t)}} \mathcal{L}(\boldsymbol{\theta}^{(t)})$$

for some learning rate schedule γ_t, we will converge to the optimum in finite time.

Gradient-Based Optimisation

Iterating

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma_t \boldsymbol{\nabla}_{\boldsymbol{\theta}^{(t)}} \mathcal{L}(\boldsymbol{\theta}^{(t)})$$

for some learning rate schedule γ_t, we will converge to the optimum in finite time.

Challenges:

- The learning rate schedule is not necessarily easy to choose.
- The gradient depends on the log-likelihood at step t.
- The log-likelihood at step t depends on the normaliser at step t.
- We need to recompute the normaliser every time the parameters change.

But this is, at least, feasible in principle.

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k
ϕ_1	A, B	(a^0, b^0)	1
		(a^0, b^1)	2
		(a^1, b^0)	3
		(a^1, b^1)	4
ϕ_2	B, C	(b^0, c^0)	5
		(b^0, c^1)	6
		(b^1, c^0)	7
		(b^1, c^1)	8
ϕ_3	C, D	(c^0, d^0)	9
		(c^0, d^1)	10
		(c^1, d^0)	11
		(c^1, d^1)	12
ϕ_4	D, A	(d^0, a^0)	13
		(d^0, a^1)	14
		(d^1, a^0)	15
		(d^1, a^1)	16

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$
ϕ_1	A, B	(a^0, b^0)	1	
		(a^0, b^1)	2	
		(a^1, b^0)	3	
		(a^1, b^1)	4	
ϕ_2	B, C	(b^0, c^0)	5	
		(b^0, c^1)	6	
		(b^1, c^0)	7	
		(b^1, c^1)	8	
ϕ_3	C, D	(c^0, d^0)	9	
		(c^0, d^1)	10	
		(c^1, d^0)	11	
		(c^1, d^1)	12	
ϕ_4	D, A	(d^0, a^0)	13	
		(d^0, a^1)	14	
		(d^1, a^0)	15	
		(d^1, a^1)	16	

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$
ϕ_1	A, B	(a^0, b^0)	1	0
		(a^0, b^1)	2	1
		(a^1, b^0)	3	0
		(a^1, b^1)	4	0
ϕ_2	B, C	(b^0, c^0)	5	
		(b^0, c^1)	6	
		(b^1, c^0)	7	
		(b^1, c^1)	8	
ϕ_3	C, D	(c^0, d^0)	9	
		(c^0, d^1)	10	
		(c^1, d^0)	11	
		(c^1, d^1)	12	
ϕ_4	D, A	(d^0, a^0)	13	
		(d^0, a^1)	14	
		(d^1, a^0)	15	
		(d^1, a^1)	16	

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$
ϕ_1	A, B	(a^0, b^0)	1	0
		(a^0, b^1)	2	1
		(a^1, b^0)	3	0
		(a^1, b^1)	4	0
ϕ_2	B, C	(b^0, c^0)	5	0
		(b^0, c^1)	6	0
		(b^1, c^0)	7	0
		(b^1, c^1)	8	1
ϕ_3	C, D	(c^0, d^0)	9	
		(c^0, d^1)	10	
		(c^1, d^0)	11	
		(c^1, d^1)	12	
ϕ_4	D, A	(d^0, a^0)	13	
		(d^0, a^1)	14	
		(d^1, a^0)	15	
		(d^1, a^1)	16	

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$
ϕ_1	A, B	(a^0, b^0)	1	0
		(a^0, b^1)	2	1
		(a^1, b^0)	3	0
		(a^1, b^1)	4	0
ϕ_2	B, C	(b^0, c^0)	5	0
		(b^0, c^1)	6	0
		(b^1, c^0)	7	0
		(b^1, c^1)	8	1
ϕ_3	C, D	(c^0, d^0)	9	0
		(c^0, d^1)	10	0
		(c^1, d^0)	11	1
		(c^1, d^1)	12	0
ϕ_4	D, A	(d^0, a^0)	13	
		(d^0, a^1)	14	
		(d^1, a^0)	15	
		(d^1, a^1)	16	

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$
ϕ_1	A, B	(a^0, b^0)	1	0
		(a^0, b^1)	2	1
		(a^1, b^0)	3	0
		(a^1, b^1)	4	0
ϕ_2	B, C	(b^0, c^0)	5	0
		(b^0, c^1)	6	0
		(b^1, c^0)	7	0
		(b^1, c^1)	8	1
ϕ_3	C, D	(c^0, d^0)	9	0
		(c^0, d^1)	10	0
		(c^1, d^0)	11	1
		(c^1, d^1)	12	0
ϕ_4	D, A	(d^0, a^0)	13	1
		(d^0, a^1)	14	0
		(d^1, a^0)	15	0
		(d^1, a^1)	16	0

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$	$f_k(a^1, b^0, c^0, d^1)$
ϕ_1	A, B	(a^0, b^0)	1	0	
		(a^0, b^1)	2	1	
		(a^1, b^0)	3	0	
		(a^1, b^1)	4	0	
ϕ_2	B, C	(b^0, c^0)	5	0	
		(b^0, c^1)	6	0	
		(b^1, c^0)	7	0	
		(b^1, c^1)	8	1	
ϕ_3	C, D	(c^0, d^0)	9	0	
		(c^0, d^1)	10	0	
		(c^1, d^0)	11	1	
		(c^1, d^1)	12	0	
ϕ_4	D, A	(d^0, a^0)	13	1	
		(d^0, a^1)	14	0	
		(d^1, a^0)	15	0	
		(d^1, a^1)	16	0	

The most common way to parameterise an MN is to construct a fixed-dimension real-valued 'feature function'.

Factor ϕ_i	$\boldsymbol{D}_i = \text{Scope}[\phi_i]$	$\text{Val}(\boldsymbol{D}_i)$	k	$f_k(a^0, b^1, c^1, d^0)$	$f_k(a^1, b^0, c^0, d^1)$
ϕ_1	A, B	(a^0, b^0)	1	0	0
		(a^0, b^1)	2	1	0
		(a^1, b^0)	3	0	1
		(a^1, b^1)	4	0	0
ϕ_2	B, C	(b^0, c^0)	5	0	1
		(b^0, c^1)	6	0	0
		(b^1, c^0)	7	0	0
		(b^1, c^1)	8	1	0
ϕ_3	C, D	(c^0, d^0)	9	0	0
		(c^0, d^1)	10	0	1
		(c^1, d^0)	11	1	0
		(c^1, d^1)	12	0	0
ϕ_4	D, A	(d^0, a^0)	13	1	0
		(d^0, a^1)	14	0	0
		(d^1, a^0)	15	0	0
		(d^1, a^1)	16	0	1

Log-Linear Parameterisation

Once we have a K-dimensional feature function f that is able to embed all possible clique assignments in $\mathbb{R}^K$, we can introduce K real-valued coefficients $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)^\top$ and parameterise the unnormalised distribution via:

$$\tilde{P}(\boldsymbol{X} = \boldsymbol{x}) \triangleq \exp\left(\sum_{k=1}^{K} \theta_k f_k(\boldsymbol{x})\right)$$

This is equivalent to having a small log-linear model predict the cells of the tabular factors. But here I use this 'global' representation which is how the textbook presents it in Section 4.2.1.

Even though I write $f_k(\boldsymbol{x})$, the kth feature is not a function of the entire assignment $\boldsymbol{X} = \boldsymbol{x}$, since f_k concerns only one of the cliques.

Under the log-linear parameterisation

$$P_\Phi(X = x; \theta) = \frac{\exp\left(\sum_{k=1}^{K} \theta_k f_k(x)\right)}{Z(\theta)}$$

Under the log-linear parameterisation

$$P_\Phi(X = x; \boldsymbol{\theta}) = \frac{\exp\left(\sum_{k=1}^{K} \theta_k f_k(x)\right)}{Z(\boldsymbol{\theta})}$$

And hence

$$\log P_\Phi(X = x; \boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k f_k(x) - \log Z(\boldsymbol{\theta})$$

This means that with a dataset $\mathcal{D}$, the log-likelihood becomes:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) =$$

Under the log-linear parameterisation

$$P_\Phi(X = x; \boldsymbol{\theta}) = \frac{\exp\left(\sum_{k=1}^{K} \theta_k f_k(x)\right)}{Z(\boldsymbol{\theta})}$$

And hence

$$\log P_\Phi(X = x; \boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k f_k(x) - \log Z(\boldsymbol{\theta})$$

This means that with a dataset $\mathcal{D}$, the log-likelihood becomes:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \left(\sum_{m=1}^{M} \sum_{k=1}^{K} \theta_k f_k(x[m])\right) - M \log Z(\boldsymbol{\theta})$$

Log-Linear Parameterisation - Gradient

The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ that we need for parameter estimation is a vector of partial derivatives, the kth coordinate of it is shown below:

$$\frac{\partial}{\partial \theta_k} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[f_k(X)] - \mathbb{E}_{\boldsymbol{\theta}}[f_k(X)]$$

Log-Linear Parameterisation - Gradient

The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ that we need for parameter estimation is a vector of partial derivatives, the kth coordinate of it is shown below:

$$\frac{\partial}{\partial \theta_k} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[f_k(X)] - \mathbb{E}_{\boldsymbol{\theta}}[f_k(X)]$$

The first term is the expected value of the kth feature estimated using **data samples**:

- This is simple to compute, the feature function is a simple lookup operation, we just compute the feature value of each data sample and average the result.

Log-Linear Parameterisation - Gradient

The gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D})$ that we need for parameter estimation is a vector of partial derivatives, the kth coordinate of it is shown below:

$$\frac{\partial}{\partial \theta_k} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[f_k(X)] - \mathbb{E}_{\boldsymbol{\theta}}[f_k(X)]$$

The first term is the expected value of the kth feature estimated using **data samples**:

- This is simple to compute, the feature function is a simple lookup operation, we just compute the feature value of each data sample and average the result.

The second term is the expected value of the kth feature under the (current) **model distribution** $P_{\Phi}(X; \boldsymbol{\theta})$.

- This is intractable in general, but Gibbs sampling offers us a way to estimate it tractably.

Summary

Learning MNs is considerably more challenging because of the global normaliser.

Since there's no analytical solution, not even in tabular form, it is convenient to work with a more general log-linear parameterisation.

We learn MNs via gradient-based optimisation, while the log-likelihood still has a global optimum, we face some computational challenges expressing (or estimating) the gradient. MCMC is one of the strategies that can come to the rescue.

Learning with Missing Data

Missing Data

Sometimes we can collect only **small datasets with complete observations, and very large datasets containing only partial (or incomplete) observations**.

For example, we may have data records for millions of patients for which symptoms/effects (e.g., cough, high temperature, headache, loss of sense of smell) were known but the cause (e.g., COVID19, the flu) was never identified (e.g., perhaps they never took a conclusive test), but only a few thousands paired effect-cause data records.

Should we discard data points that are 'incomplete' in this sense?

Missing Data

Sometimes we can collect only **small datasets with complete observations, and very large datasets containing only partial (or incomplete) observations**.

For example, we may have data records for millions of patients for which symptoms/effects (e.g., cough, high temperature, headache, loss of sense of smell) were known but the cause (e.g., COVID19, the flu) was never identified (e.g., perhaps they never took a conclusive test), but only a few thousands paired effect-cause data records.

Should we discard data points that are 'incomplete' in this sense?

No, at least not if we have a probabilistic model of all rvs involved!

Missing Data

Sometimes we can collect only **small datasets with complete observations, and very large datasets containing only partial (or incomplete) observations**.

For example, we may have data records for millions of patients for which symptoms/effects (e.g., cough, high temperature, headache, loss of sense of smell) were known but the cause (e.g., COVID19, the flu) was never identified (e.g., perhaps they never took a conclusive test), but only a few thousands paired effect-cause data records.

Should we discard data points that are 'incomplete' in this sense?

No, at least not if we have a probabilistic model of all rvs involved!

A PGM **is** a mechanism to, amongst many other things, infer the distribution of unassigned/unobserved rvs $\boldsymbol{U}$ given assigned/observed evidence: $P(\boldsymbol{U}|\boldsymbol{E} = \boldsymbol{e})$.

Statistical Influence

Learning from incomplete observations can be very useful, as the observed data likely still carry plenty of statistically useful information due to paths/trails of influence in PGMs.

Strategy

1. Initialise the model (for example, estimate the parameters using all available complete data).
2. Use the current model to ‘complete’ the data that’s missing: assign the missing variables probabilistically.
3. Re-estimate the parameters using all of the data (possibly, weighing down the ‘completed’ part).
4. Repeat from 2 until convergence or for a maximum number of steps.

Strategy

1. Initialise the model (for example, estimate the parameters using all available complete data).
2. Use the current model to ‘complete’ the data that’s missing: assign the missing variables probabilistically.
3. Re-estimate the parameters using all of the data (possibly, weighing down the ‘completed’ part).
4. Repeat from 2 until convergence or for a maximum number of steps.

Common strategis for step 3:

- use the MAP assignment (max-product VE)
- sample an assignment (or many) using MCMC

The End?

Topics

- Bayesian networks: PGMs based on hierarchically organised local probabilistic models (CPDs).
- Markov networks: PGMs based on globally normalised product of factors.
- Exact inference: sum-product and max-product VE to solve complex probability queries.
- Approximate inference: Gibbs sampling to draw samples from the model.
- Learning: MLE for BNs and a ‘preview’ of MLE for MNs.

ILOs

- Factorise and parameterise models.
- Reason about influence and conditional independence.
- Manipulate factors and analyse factor operations.
- Solve complex probability queries with and without evidence.
- Draw samples from a model with or without evidence and estimate expectations.
- Estimate the parameters of BNs from data.
- Estimate the parameters of MNs from data.
- Algorithms for learning with missing data.
- Meet a few important model families: naive Bayes and (hidden) Markov models, pairwise MNs and conditional random fields (CRFs).
- Implement all of the above in Python.

The End

Sort of

The textbook has over 1000 pages and it was written in 2009, so I expect this not to be the finish line for PGMs in your career.

But, **we covered the core of the PGM framework**, and **you should be ready to dive into more advanced topics and applications with a strong foundation**.

Some courses that will surely build upon the knowledge you developed

- 2nd year: NLP, CV, cognitive modelling (I suspect)
- 3rd year: text, speech and dialogue, ML for structured data.
- MSc AI: practically *all of it*; advanced PGM topics are covered in ML2 and DL2.

Enjoy the rest of the programme, and see you in block 4!

References

[1] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.