Introduction to graphical models and generative models

theory and examples

Pierre-Alexandre Mattei

Menu for this lecture

1. Recap on the course: pratical details + Why graphical models? Why generative models?
2. Recap on probability theory
3. Basics of graph theory
4. Directed graphical models, theory and examples
5. Undirected graphical models (if we have time)

0

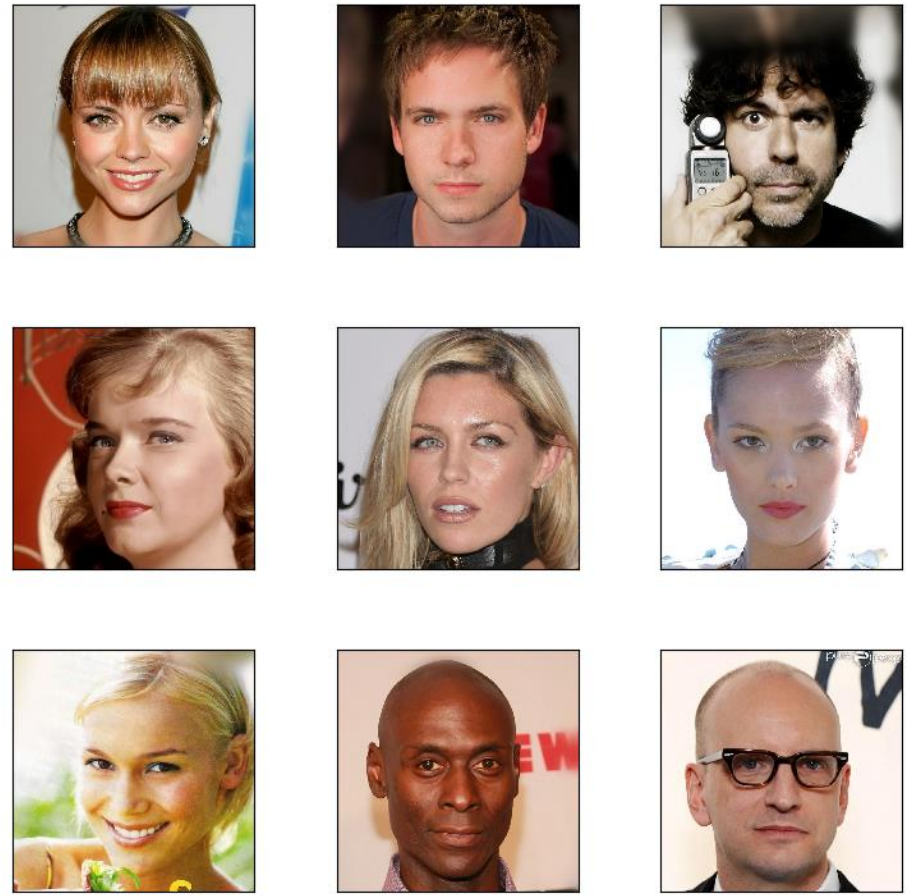
What is this course?

Recap on the course

- Two lecturers: Pierre Latouche (Université Clermont Auvergne) & Pierre-Alexandre Mattei (Inria, Univ. Côte d'Azur)
- One TA: Rémi Khellaf
- ‘‘Hybrid’’ lectures on zoom and at ENS Saclay
- Evaluation: project on one or several papers (poster session + report)
- **Goal: learn how to design and train graphical models, from simple models to deep generative models**

What is generative modelling?

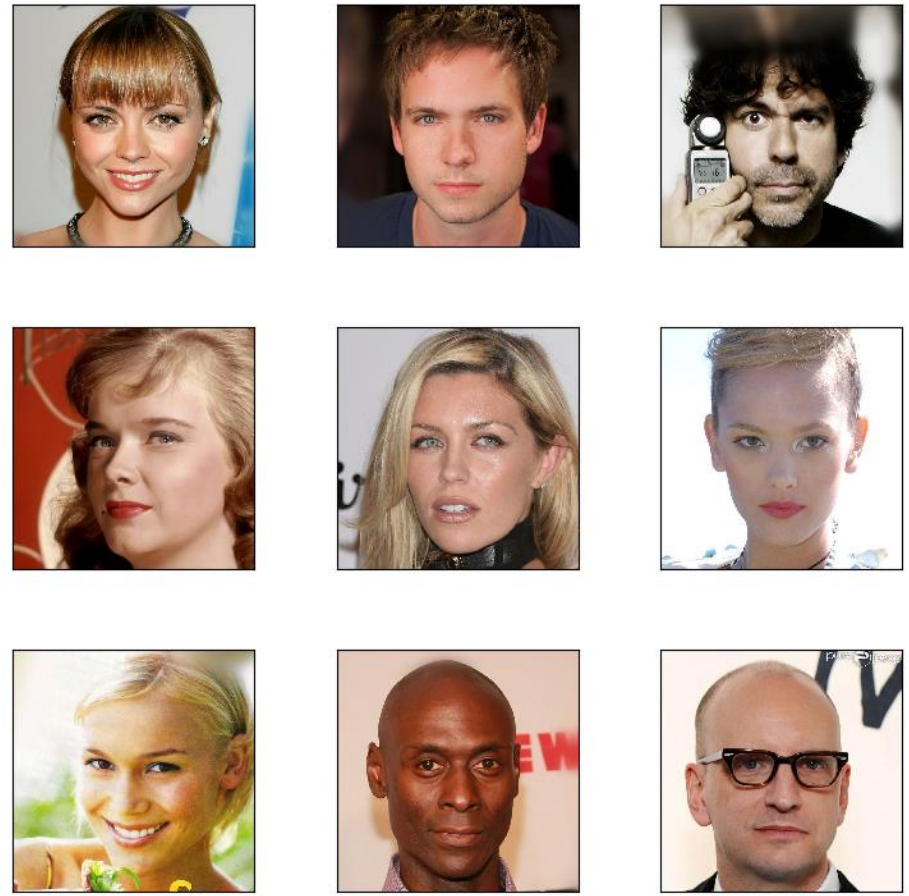
- We observe some data, e.g., the CelebA dataset
- Goal: learn approximation of the **data distribution $p(\mathcal{D})$**
- **But why would we want that?**



Images from: https://www.tensorflow.org/datasets/catalog/celeb_a_hq

What is generative modelling?

- We observe some data, e.g., the CelebA dataset
- Goal: learn approximation of the **data distribution $p(\mathcal{D})$**
- **But why would we want that?**
  - ➢ **Because it solves everything!**



Images from: https://www.tensorflow.org/datasets/catalog/celeb_a_hq

A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Classification**

  $\mathcal{D} = (x, y)$ with y discrete

  $p(y|x)$ gives you the **best possible classifier!**

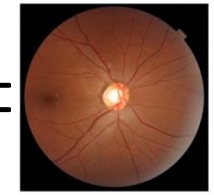
A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Classification**

  $\mathcal{D} = (x, y)$ with y discrete

  $p(y|x)$ gives you the **best possible classifier!**

$$p(y = \text{retinopathy} | x = \text{  })$$

Data from https://isbi.deepdr.org/data.html

A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Regression/forecasting**
  - $\mathcal{D} = (x, y)$ with $y \in \mathbb{R}$
  - $\mathbb{E}[y|x]$ gives you the **best possible regression function!**
  - $p(y|x)$ gives you **perfect prediction intervals!**

A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Regression/forecasting**
  - $\mathcal{D} = (x, y)$ with $y \in \mathbb{R}$
  - $\mathbb{E}[y|x]$ gives you the **best possible regression function!**
  - $p(y|x)$ gives you **perfect prediction intervals!**

$$p(y \in [a, b]|x)$$

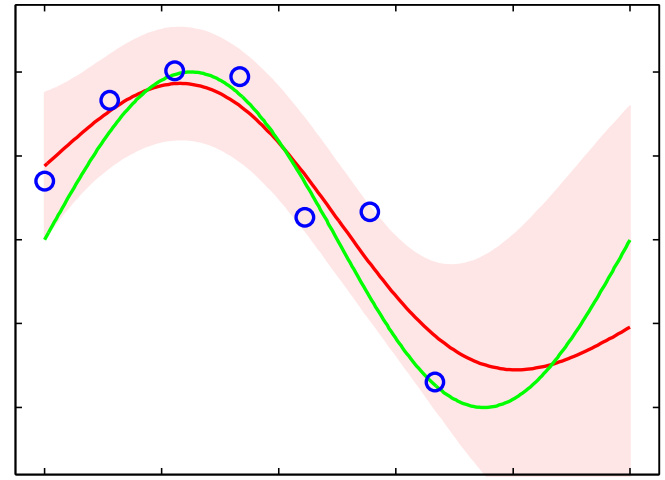


Fig. 6.8 from Bishop, Pattern Recognition and Machine Learning, 2006

A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Missing data imputation**

  $\mathcal{D} = (x^{obs}, x^{miss})$

  $\mathbb{E}[x^{miss}|x^{obs}]$ gives the **best possible simple imputation!**

  $p(x^{miss}|x^{obs})$ gives the **best possible multiple imputations!**

A probability distribution solves everything!

If we have properly learned $p(\mathcal{D})$, we can perform all sort of tasks:

- **Missing data imputation**

  $\mathcal{D} = (x^{obs}, x^{miss})$

  $\mathbb{E}[x^{miss}|x^{obs}]$ gives the **best possible simple imputation!**

  $p(x^{miss}|x^{obs})$ gives the **best possible multiple imputations!**

Fig. 1 from Lugmayr et al., RePaint: Inpainting using Denoising Diffusion Probabilistic Models, CVPR 2022

A probability distribution solves everything!

Sometimes, we model more than just the data, we also add **latent variables**

- **Clustering**

  $\mathcal{D} = (x, y)$ with y discrete and latent

  $p(y|x)$ gives you **model-based clusters!**

A probability distribution solves everything!

Sometimes, we model more than just the data, we also add **latent variables**

- **Clustering**
  $\mathcal{D} = (x, y)$ with y discrete and latent
  $p(y|x)$ gives you **model-based clusters!**

- **Dimension-reduction/representation learning**
  $\mathcal{D} = (x, y)$ with y continuous and latent
  $\mathbb{E}[y|x]$ gives you **probabilistic dimension-reduction!**

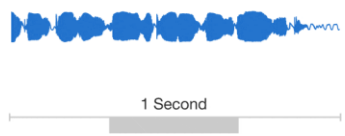
Not only $p(\mathcal{D})$ solves everything, but we can sample cool new stuff from it!

$$p(x|t = \text{“A head of broccoli complaining about the weather”})$$

Sample from OpenAI's Dall-E 2, prompt by @TheRealAdamG

It’s not just about pretty images and texts!

WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



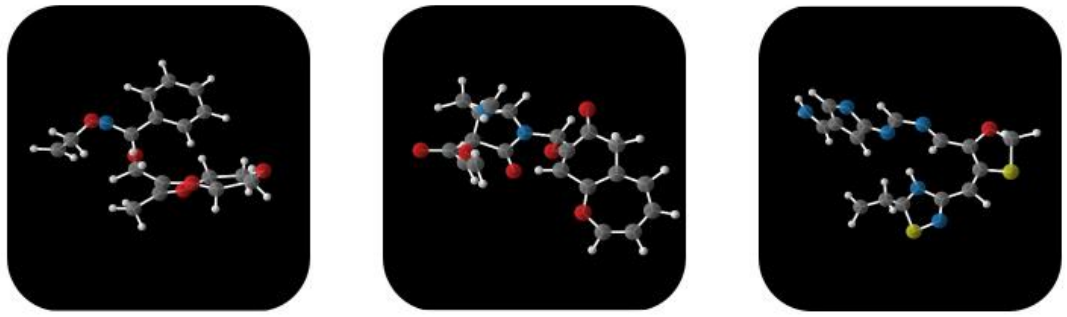
Aäron van den Oord Sander Dieleman Heiga Zen†
Karen Simonyan Oriol Vinyals Alex Graves
Nal Kalchbrenner Andrew Senior Koray Kavukcuoglu

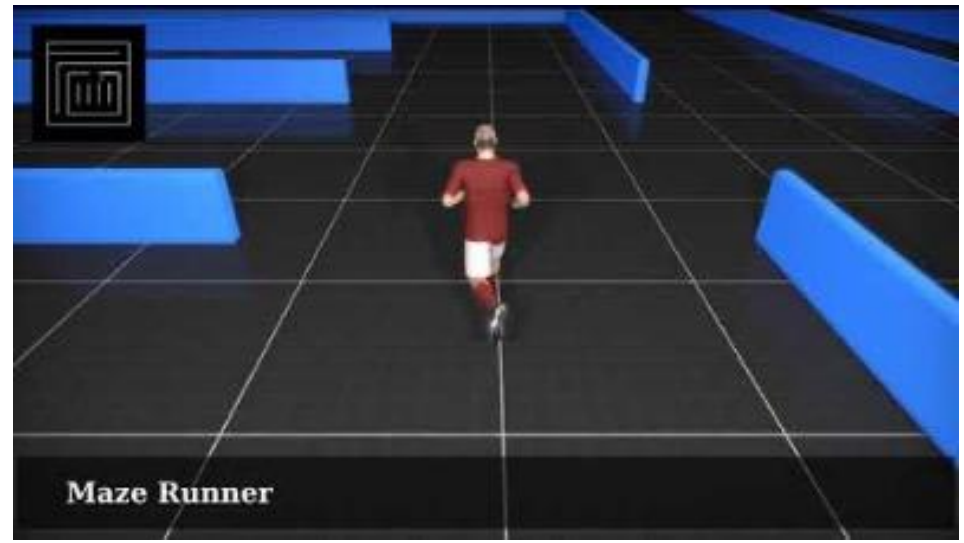
Character Controllers Using Motion VAEs

HUNG YU LING, University of British Columbia, Canada
FABIO ZINNO, Electronic Arts Vancouver, Canada
GEORGE CHENG, Electronic Arts Vancouver, Canada
MICHIEL VAN DE PANNE, University of British Columbia, Canada

Equivariant Diffusion for Molecule Generation in 3D

Emiel Hoogeboom [*1] Victor Garcia Satorras [*1] Clément Vignac [*2] Max Welling [1]





What’s the catch?

- Generative models can solve virtually all machine learning tasks, so **why not do everything in a generative way?**

- This sounds a bit too optimistic, **is there a catch?**

The catch is that generative modelling is hard!

- Often, **generative modelling is much harder than directly attacking the task** at hand (e.g. classification, regression, imputation).

The catch is that generative modelling is hard!

- Often, **generative modelling is much harder than directly attacking the task** at hand (e.g. classification, regression, imputation).

When solving a given problem, try to avoid solving a more general problem as an intermediate step.

Vladimir Vapnik, The Nature of Statistical Learning Theory, 2000, Section 1.9

The main recipe for designing generative models is to use graphical models.

General motivation for graphical modelling

- Typcally, we observe data with **mutiple modalities**, and potentially **latent factors**

General motivation for graphical modelling

- Typcally, we observe data with **mutiple modalities**, and potentially **latent factors**
- The **concrete tasks** we want to solve can be rephrased as making **probabilistic statements** about the value of these factors or modalities.

General motivation for graphical modelling

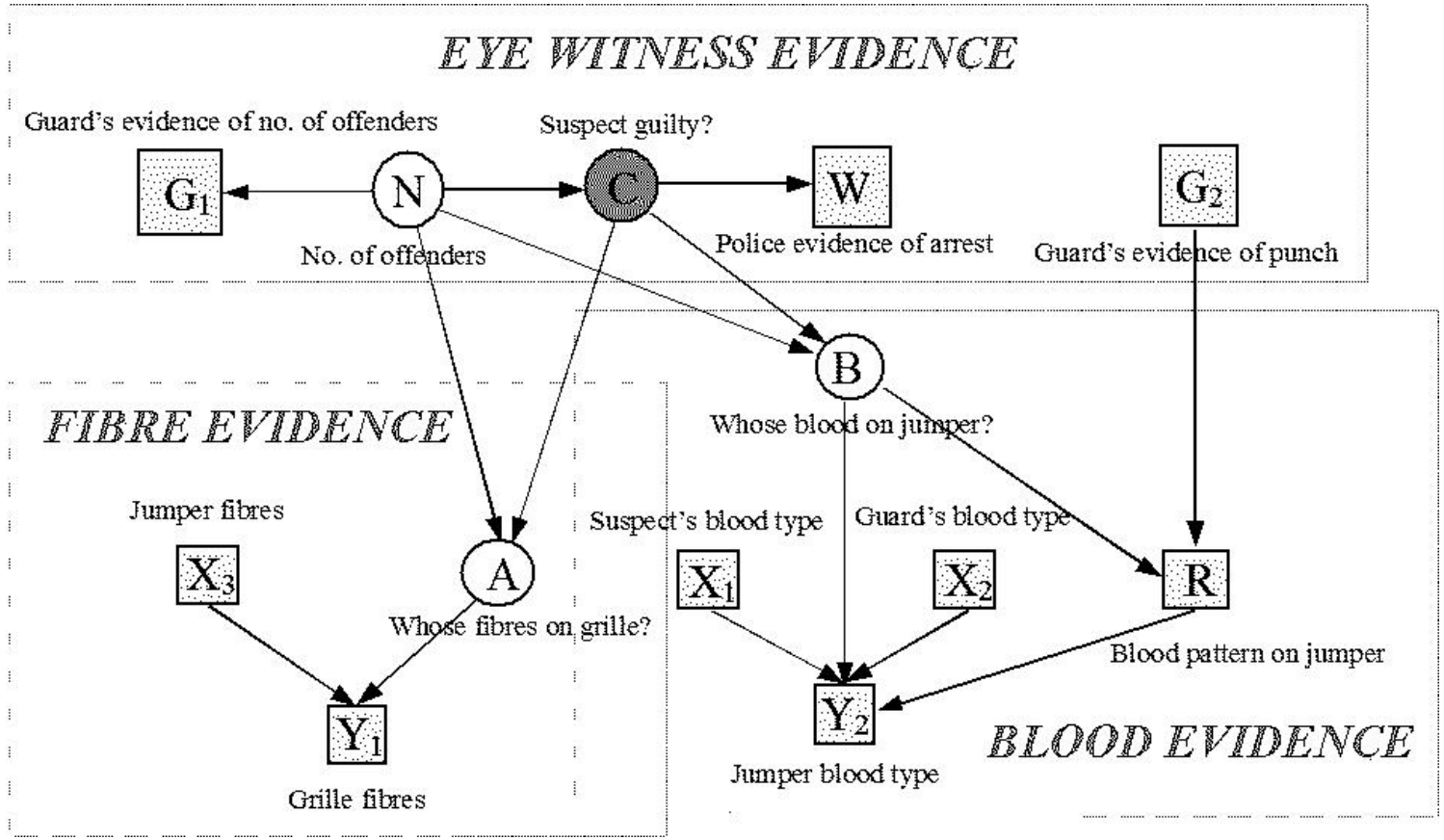
- Typcally, we observe data with **mutiple modalities**, and potentially **latent factors**
- The **concrete tasks** we want to solve can be rephrased as making **probabilistic statements** about the value of these factors or modalities.
- By probabilisitic statement, I mean e.g. computing $\mathbb{E}[y|x], p(y|x)$, **or any of the examples from the previous slides.**

General motivation for graphical modelling

- Typcally, we observe data with **mutiple modalities**, and potentially **latent factors**
- The **concrete tasks** we want to solve can be rephrased as making **probabilistic statements** about the value of these factors or modalities.
- By probabilisitic statement, I mean e.g. computing $\mathbb{E}[y|x], p(y|x)$, **or any of the examples from the previous slides.**
- In particular, we are interested by **conditional distributions**

What graphical models? An old-school example

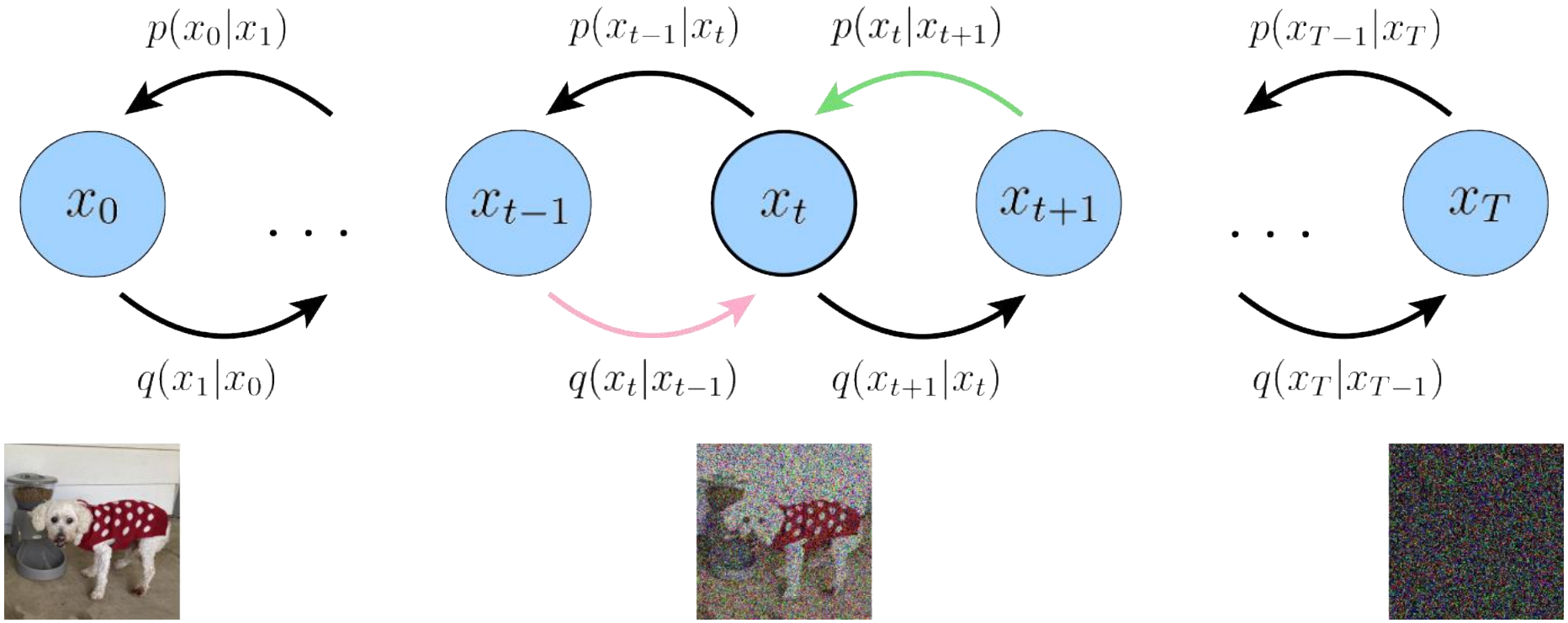
- We will build **graphs** where each feature correspond to a modality or a factor



Dawid and Evett, Using a graphical method to assist the evaluation of complicated patterns of evidence. Journal of Forensic Sciences, 1997

What graphical models? A new-school example

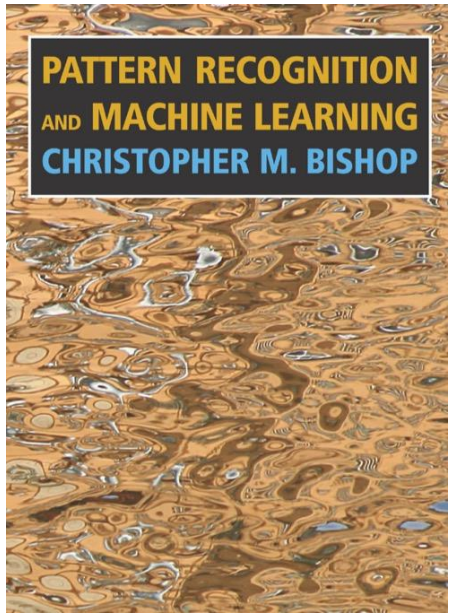
- We will build **graphs** where each feature correspond to a modality or a factor



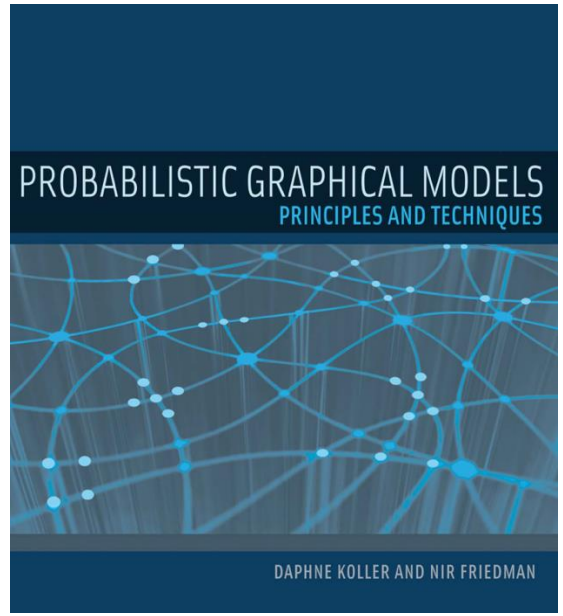
Luo, Understanding Diffusion Models: A Unified Perspective, arXiv preprint, 2022

Some useful references

As much as possible, these lectures will be self-contained. More details, including proofs of the results we will not prove in class, can be found in a few key references. I will refer to them sometimes. I will also sometimes refer as [MVA17] to the previous lecture notes of this course.



[Bishop]



[PGM]

Statistical Science
2004, Vol. 19, No. 1, 140–155
DOI 10.1214/088342304000000026

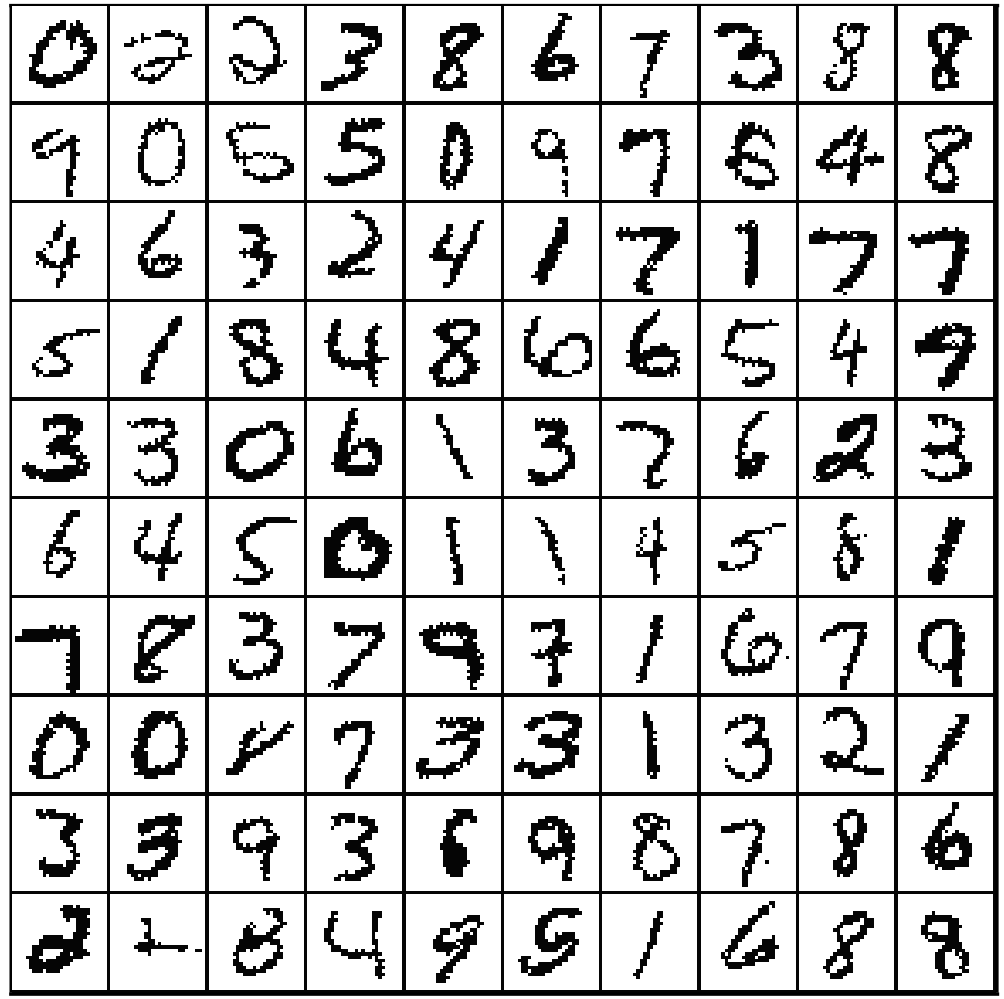
Graphical Models

Michael I. Jordan

[Jordan]

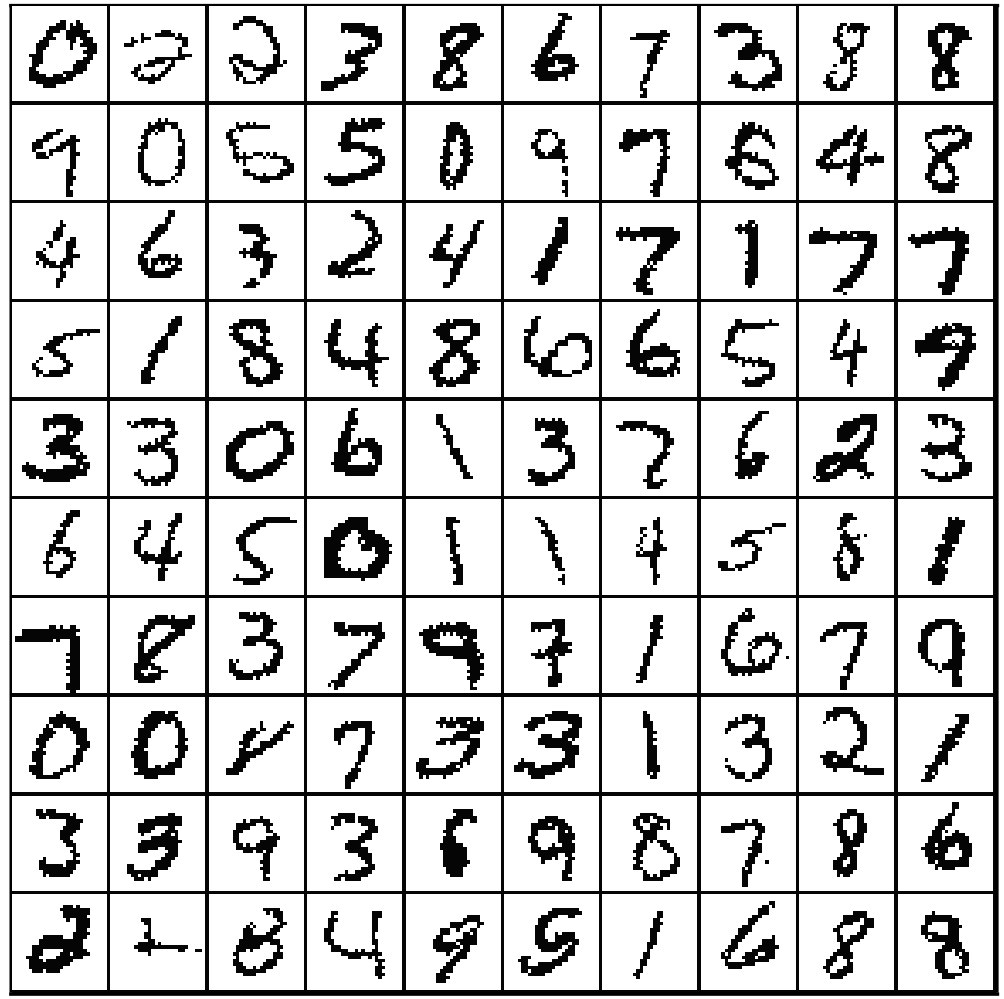
Recurring « fil rouge » example

- We will often consider as a working example a **binary version of MNIST**
- $x_1, ..., x_n \in \mathcal{X} \quad \mathcal{X} = \{0,1\}^{28\times 28}$
- $y_1, ..., y_n \in \{0, \ldots, 9\}$



Recurring « fil rouge » example

- We will often consider as a working example a **binary version of MNIST**
- $x_1, ..., x_n \in \mathcal{X} \quad \mathcal{X} = \{0,1\}^{28\times 28}$
- $y_1, ..., y_n \in \{0, \ldots, 9\}$
- Binarised by Hugo Larochelle
- Sometimes called « statical binary MNIST » in VAE papers



1

Recap on probability theory

A quick recap mostly to fix notations

- We study a random variable $X = (X_1, ..., X_d) \in \mathcal{X}_1 \times ... \times \mathcal{X}_d$

$\mathbb{P}(\theta|x)$

A quick recap mostly to fix notations

- We study a random variable $X = (X_1, ..., X_d) \in \mathcal{X}_1 \times ... \times \mathcal{X}_d$
- Its density with respect to a base measure (typically the Lebesgue measure if the data are continuous, the counting measure in the discrete case, or the product thereof in the heterogeneous case) is denoted by

$$p(x) = p(x_1, ..., x_d)$$

A quick recap mostly to fix notations

- We study a random variable $X = (X_1, ..., X_d) \in \mathcal{X}_1 \times ... \times \mathcal{X}_d$
- Its density with respect to a base measure (typically the Lebesgue measure if the data are continuous, the counting measure in the discrete case, or the product thereof in the heterogeneous case) is denoted by

$$p(x) = p(x_1, ..., x_d)$$

- In the discrete case, this is simply

$$p(x) = \mathbb{P}(X = x) = \mathbb{P}(X_1 = x_1, ..., X_d = x_d)$$

Marginal/conditional distributions

- The density of all coordinates is called the **joint density** $p(x) = p(x_1, \ldots, x_d)$

Marginal/conditional distributions

- The density of all coordinates is called the **joint density** $p(x) = p(x_1, ..., x_d)$

- We can recover the **marginal density** of any subset of the coordinates by integrating

$$\forall A \subset \{1, ..., d\},\ p(x_A) = \int p(x_A, x_{A^c}) dx_{A^c}$$

Marginal/conditional distributions

- The density of all coordinates is called the **joint density** $p(x) = p(x_1, ..., x_d)$

- We can recover the **marginal density** of any subset of the coordinates by integrating

$$\forall A \subset \{1, ..., d\},\ p(x_A) = \int p(x_A, x_{A^c}) dx_{A^c}$$

- For two disjoint subsets , the **conditional density** is given by

$$p(x_A | x_B) = \frac{p(x_A, x_B)}{p(x_B)}$$

Parametrised distributions

- We generally want our distributions to be **learnable**, i.e. to depend on **unknown parameters**.
- If $p(x)$ depends on some parameters θ, then we often denote it $\boldsymbol{p_\theta(x)}$.
- We may also denote it $p(x|\theta)$. This is mostly useful for Bayesian models, for which θ is treated as a random variable.

Parametrised distributions

- We generally want our distributions to be **learnable**, i.e. to depend on **unknown parameters**.
- If $p(x)$ depends on some parameters θ, then we often denote it $\boldsymbol{p_\theta(x)}$.
- We may also denote it $p(x|\theta)$. This is mostly useful for Bayesian models, for which θ is treated as a random variable.
- Parameters are then generally learned via **maximum-likelihood**

$$\hat{\theta}_{\mathrm{ML}} \in \operatorname{argmax} \log p_\theta(x)$$

Parametrised distributions

- We generally want our distributions to be **learnable**, i.e. to depend on **unknown parameters**.
- If $p(x)$ depends on some parameters θ, then we often denote it $\boldsymbol{p_\theta(x)}$.
- We may also denote it $p(x|\theta)$. This is mostly useful for Bayesian models, for which θ is treated as a random variable.
- Parameters are then generally learned via **maximum-likelihood**

x is the whole dataset here

$$\hat{\theta}_{\mathrm{ML}} \in \operatorname{argmax} \log p_\theta(x)$$

A basic example of parametrised distribution for binary MNIST

- Let’s go back to our binary images, that live in $\mathcal{X} = \{0, 1\}^{28 \times 28}$
- A basic model would be to assume that each pixel corresponds to a different coin flip!

The density function of a Bernoulli/coin flip

The density of $\mathcal{B}(\pi)$ is defined as $p_\pi(x) = \begin{cases} \pi \text{ if } x = 1 \\ 1 - \pi \text{ if } x = 0 \end{cases}$

There another (sometimes more convenient) way to write it:

$$p_\pi(x) = \pi^x (1 - \pi)^{1-x}$$

The density function of a Bernoulli/coin flip

The density of $\mathcal{B}(\pi)$ is defined as $p_\pi(x) = \begin{cases} \pi \text{ if } x = 1 \\ 1 - \pi \text{ if } x = 0 \end{cases}$

There another (sometimes more convenient) way to write it:

$$p_\pi(x) = \pi^x (1 - \pi)^{1-x}$$

Minus the log of this is called the **cross-entropy loss** between x and π

$$\mathrm{CE}(x, \pi) = -\log p_\pi(x) = -x \log \pi - (1 - x) \log(1 - \pi)$$

A basic example of parametrised distribution for binary MNIST

- Let's go back to our binary images, that live in $\mathcal{X} = \{0,1\}^{28\times 28}$
- A basic model would be to assume that each pixel corresponds to a different coin flip!

$$x_{ij} \sim_{\text{iid}} \mathcal{B}(\theta_j) \text{ with } \theta_j \in (0,1) \text{ for } i \leq n, j \leq 28 \times 28$$

A basic example of parametrised distribution for binary MNIST

- Let's go back to our binary images, that live in $\mathcal{X} = \{0,1\}^{28\times 28}$
- A basic model would be to assume that each pixel corresponds to a different coin flip!

$$x_{ij} \sim_{\text{iid}} \mathcal{B}(\theta_j) \text{ with } \theta_j \in (0,1) \text{ for } i \leq n, j \leq 28 \times 28$$

- This may be rewritten

$$p_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i) = \prod_{i=1}^{n} \prod_{j=1}^{28\times 28} \mathcal{B}(x_{ij}|\theta_j)$$

A basic example of parametrised distribution for binary MNIST

- Let's go back to our binary images, that live in $\mathcal{X} = \{0,1\}^{28\times 28}$
- A basic model would be to assume that each pixel corresponds to a different coin flip!

$$x_{ij} \sim_{\text{iid}} \mathcal{B}(\theta_j) \text{ with } \theta_j \in (0,1) \text{ for } i \leq n, j \leq 28 \times 28$$

- This may be rewritten

$$p_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_\theta(x_i) = \prod_{i=1}^{n} \prod_{j=1}^{28\times 28} \mathcal{B}(x_{ij}|\theta_j)$$

- **Q:** Why is this model not great?

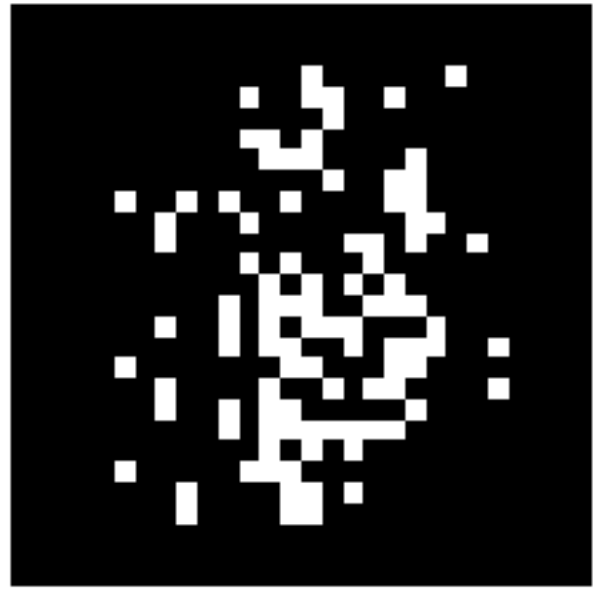
Why is this model not great?

The MLE is just $\hat{\theta}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$

Why is this model not great?

The MLE is just $\hat{\theta}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$

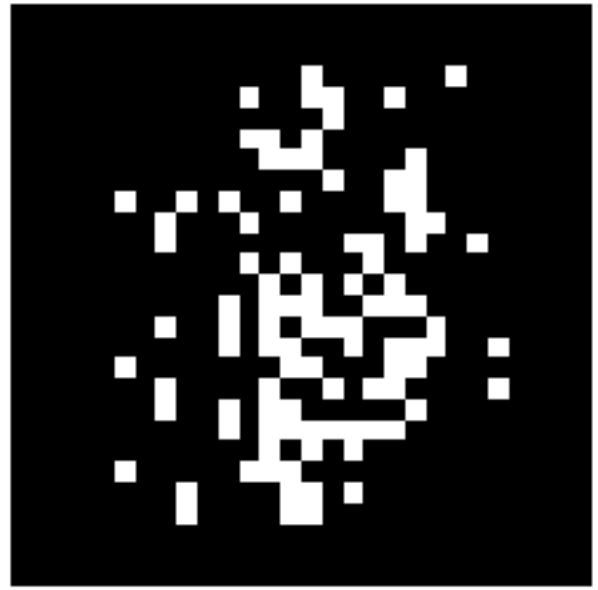
Here is a sample from the model…



Why is this model not great?

The MLE is just $\hat{\theta}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$

Here is a sample from the model…



Real pixels are not independent!

A first factorisation: the chain rule

- Another way to write the conditional density definition is

$$p(x_A, x_B) = p(x_A|x_B)p(x_B)$$

- This sort of decomposition of a joint distribution into smaller-dimensional distributions is called a **factorisation.**

A first factorisation: the chain rule

- Another way to write the conditional density definition is

$$p(x_A, x_B) = p(x_A|x_B)p(x_B)$$

- This sort of decomposition of a joint distribution into smaller-dimensional distributions is called a **factorisation.**

- Applying the above formula many times gives the **chain rule**

$$p(x) = p(x_1)p(x_2|x_1)...p(x_{d-1}|x_1, ..., x_{d-2})p(x_d|x_1, ..., x_{d-1})$$

A first factorisation: the chain rule

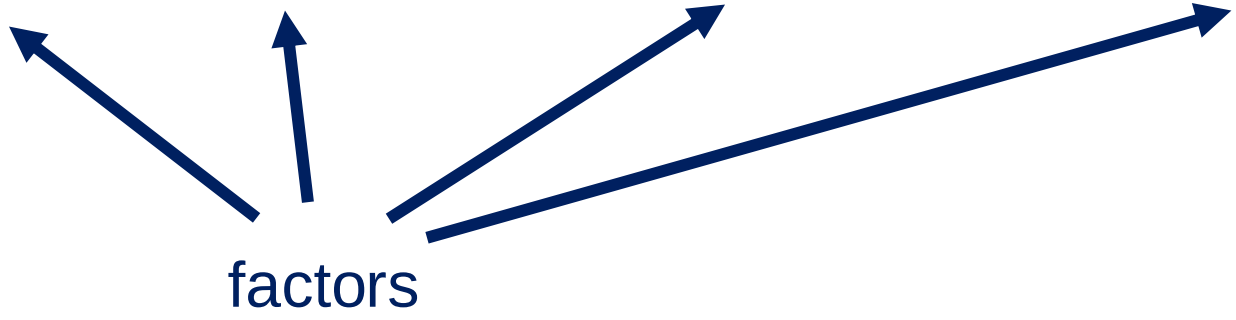
- Another way to write the conditional density definition is

$$p(x_A, x_B) = p(x_A|x_B)p(x_B)$$

- This sort of decomposition of a joint distribution into smaller-dimensional distributions is called a **factorisation.**

- Applying the above formula many times gives the **chain rule**

$$p(x) = p(x_1)p(x_2|x_1)...p(x_{d-1}|x_1, ..., x_{d-2})p(x_d|x_1, ..., x_{d-1})$$



A first factorisation: the chain rule

- Another way to write the conditional density definition is

$$p(x_A, x_B) = p(x_A|x_B)p(x_B)$$

- This sort of decomposition of a joint distribution into smaller-dimensional distributions is called a **factorisation.**

- Applying the above formula many times gives the **chain rule**

$$p(x) = p(x_1)p(x_2|x_1)...p(x_{d-1}|x_1, ..., x_{d-2})p(x_d|x_1, ..., x_{d-1})$$

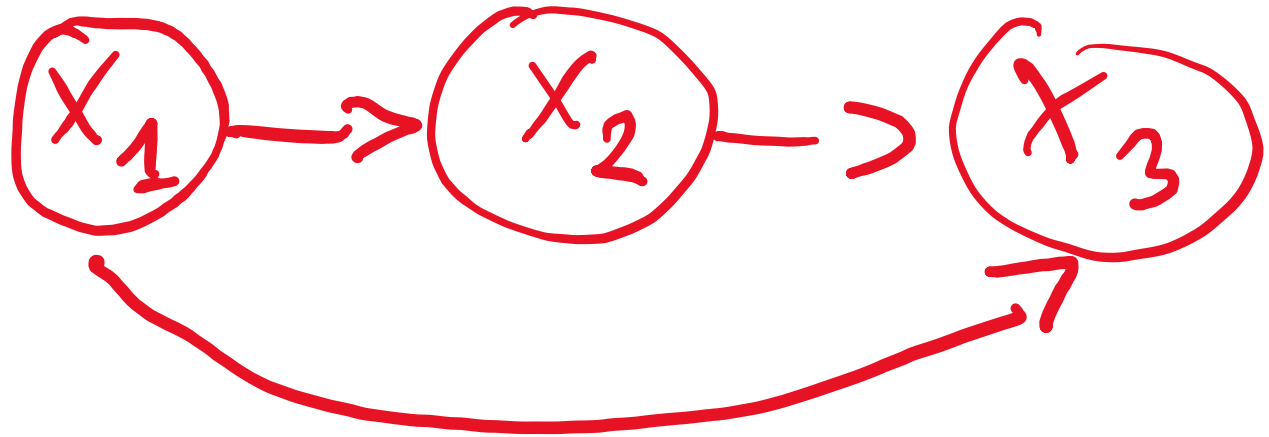
- **Factorisations** like this often lead to complex computations. **Graphical modelling provides a formal setting for analysing such factorisations** using graph theory. This leads to less equations, and more drawings!

How do we draw the chain rule?

The chain rule

$$p(x) = p(x_1)p(x_2|x_1)...p(x_{d-1}|x_1, ..., x_{d-2})p(x_d|x_1, ..., x_{d-1})$$

corresponds to the following graphical model for $d = 3$ (**we will see why later**)



Independence as another kind of factorisation

- Two variables are **independent** when $p(x_1, x_2) = p(x_1)p(x_2)$

- This is noted $x_1 \perp\!\!\!\perp x_2$

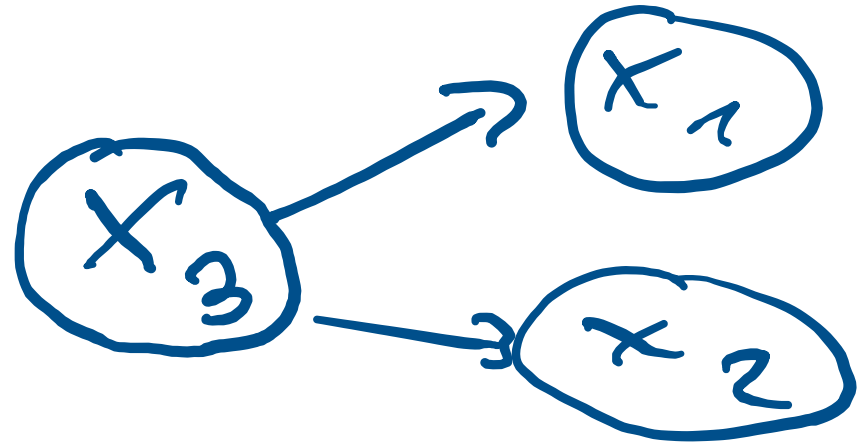
- This is the **simplest form of factorisation**, it corresponds to the graphical model

Conditional independence and how to draw it

- Two variables x_1, x_2 are conditinally **independent given** x_3 when

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

- This is noted $x_1 \perp\!\!\!\perp x_2|x_3$ and corresponds to the graphical model

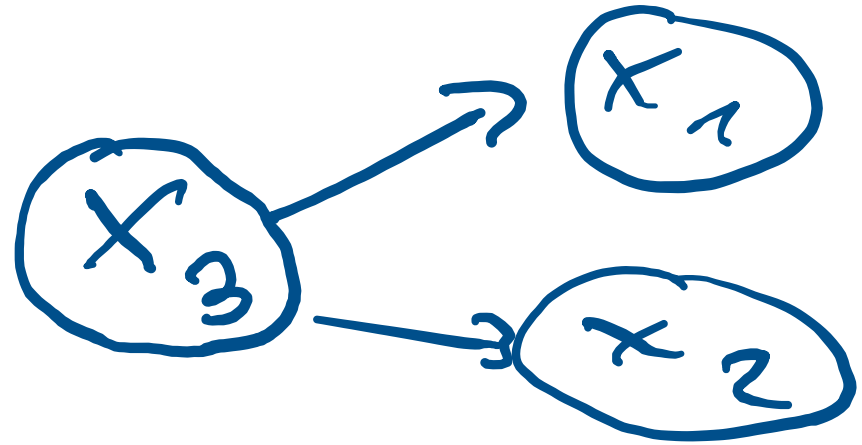


Conditional independence and how to draw it

- Two variables x_1, x_2 are conditinally **independent given** x_3 when

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

- This is noted $x_1 \perp\!\!\!\perp x_2|x_3$ and corresponds to the graphical model



- **How do we come up with these graphs? Why are they useful? That's the subject of today's lecture.**

Probabilisitic dependence and functional dependence

- In maths, another notion of dependence is functional dependence: **two quantities are functionnally dependent is there is a function that transorm one into the other**.

- Is there a link between functional dependence and probabilistic dependence? **Yes!**

Probabilisitic dependence and functional dependence

- In maths, another notion of dependence is functional dependence: **two quantities are functionnally dependent is there is a function that transorm one into the other**.

- Is there a link between functional dependence and probabilistic dependence? **Yes!**

Proposition. *Let A and B be two disjoint subsets of $\{1, ..., d\}$. If $p(x_A|x_B)$ is only functionally dependent on x_A and x_C, where C is a subset of B, i.e. if there exists a function f such that*

$$p(x_A|x_B) = f(x_A, x_C),$$

then it is equivalent to condition on x_B or x_C:

$$p(x_A|x_B) = p(x_A|x_C).$$

Proof. Left as an exercise. You can use the definition of conditional density, and marginalise out the variables that are not relevant (i.e. $B \setminus C$). □

2

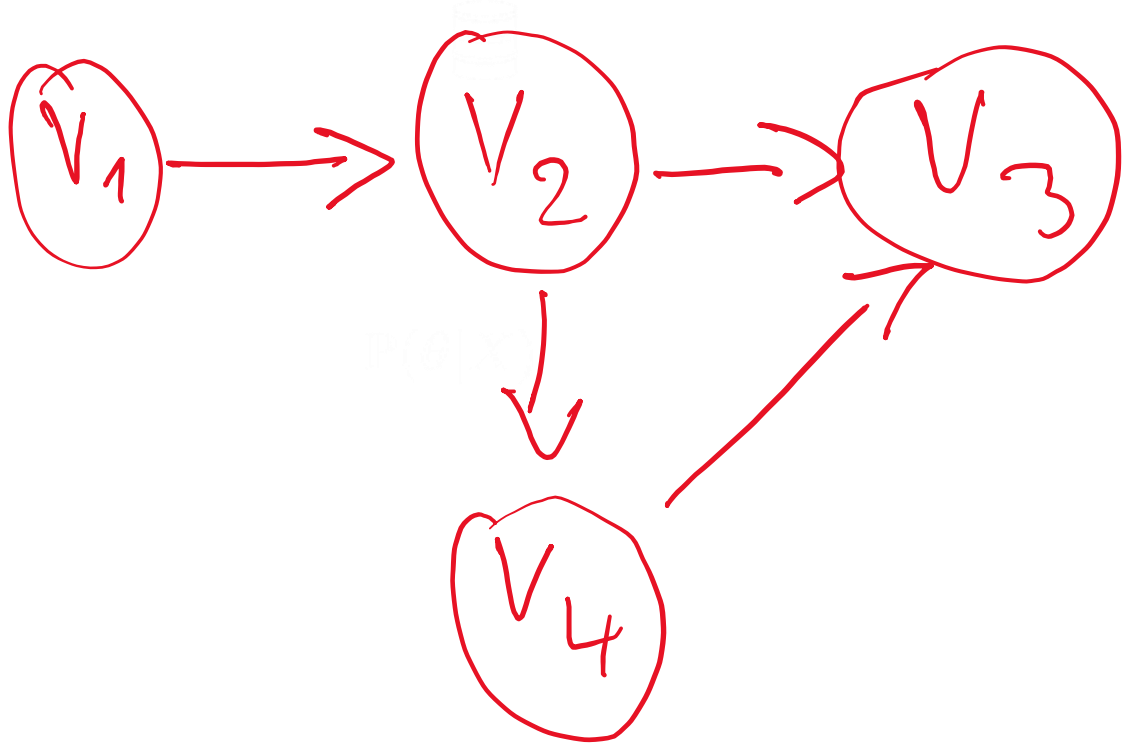
Basics of graph theory

What's a graph, mathematically?

Definition (directed graph)**.** *A* graph *is a pair* $G = (V, E)$ *comprising a set* V *of* ***vertices or nodes*** *together with a set* $E \subset V \times V$ *of* ***edges or arcs****.*

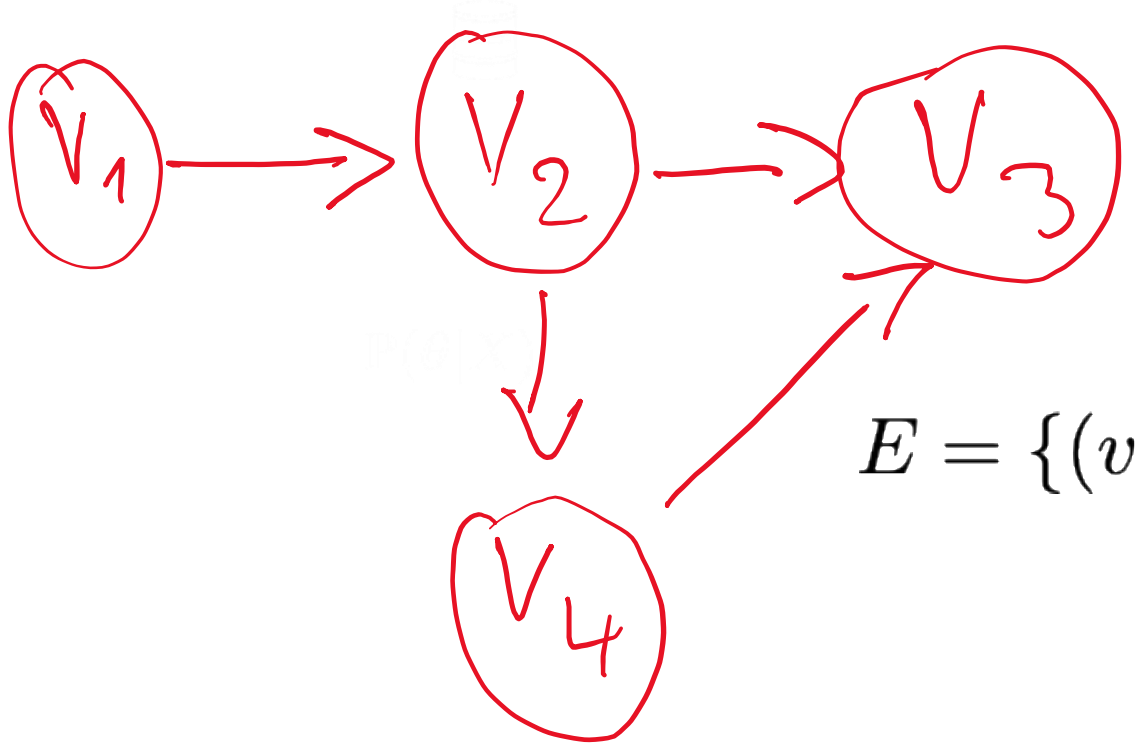
What's a graph, mathematically?

Definition (directed graph)**.** *A* graph *is a pair* $G = (V, E)$ *comprising a set* V *of* ***vertices or nodes*** *together with a set* $E \subset V \times V$ *of* ***edges or arcs****.*



What's a graph, mathematically?

Definition (directed graph)**.** *A* graph *is a pair* $G = (V, E)$ *comprising a set* V *of* ***vertices or nodes*** *together with a set* $E \subset V \times V$ *of* ***edges or arcs****.*



$$V = \{v_1, v_2, v_3, v_4\}$$

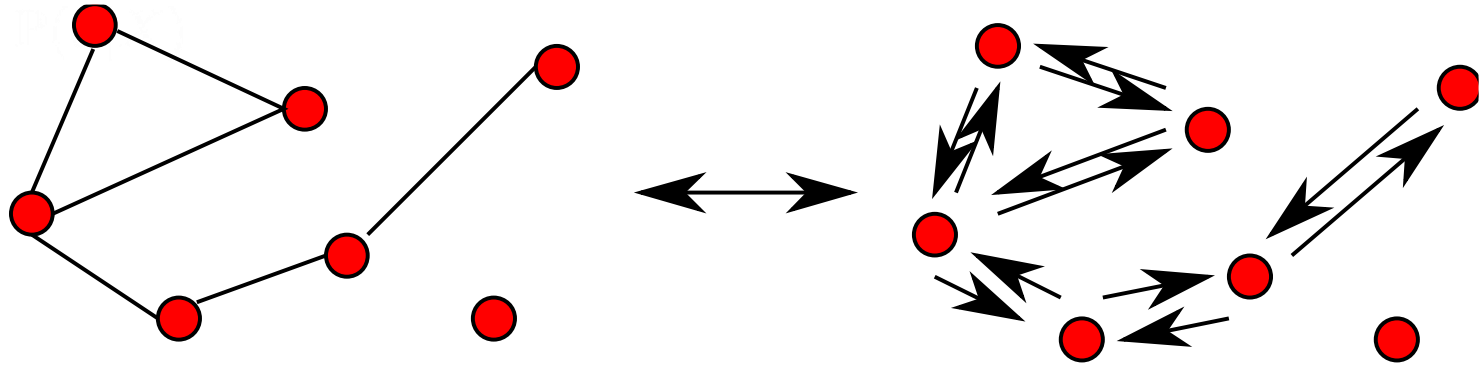
$$E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_4, v_3)\}$$

What's a graph, mathematically?

Definition (directed graph)**.** *A* graph *is a pair* $G = (V, E)$ *comprising a set* V *of* ***vertices or nodes*** *together with a set* $E \subset V \times V$ *of* ***edges or arcs****.*

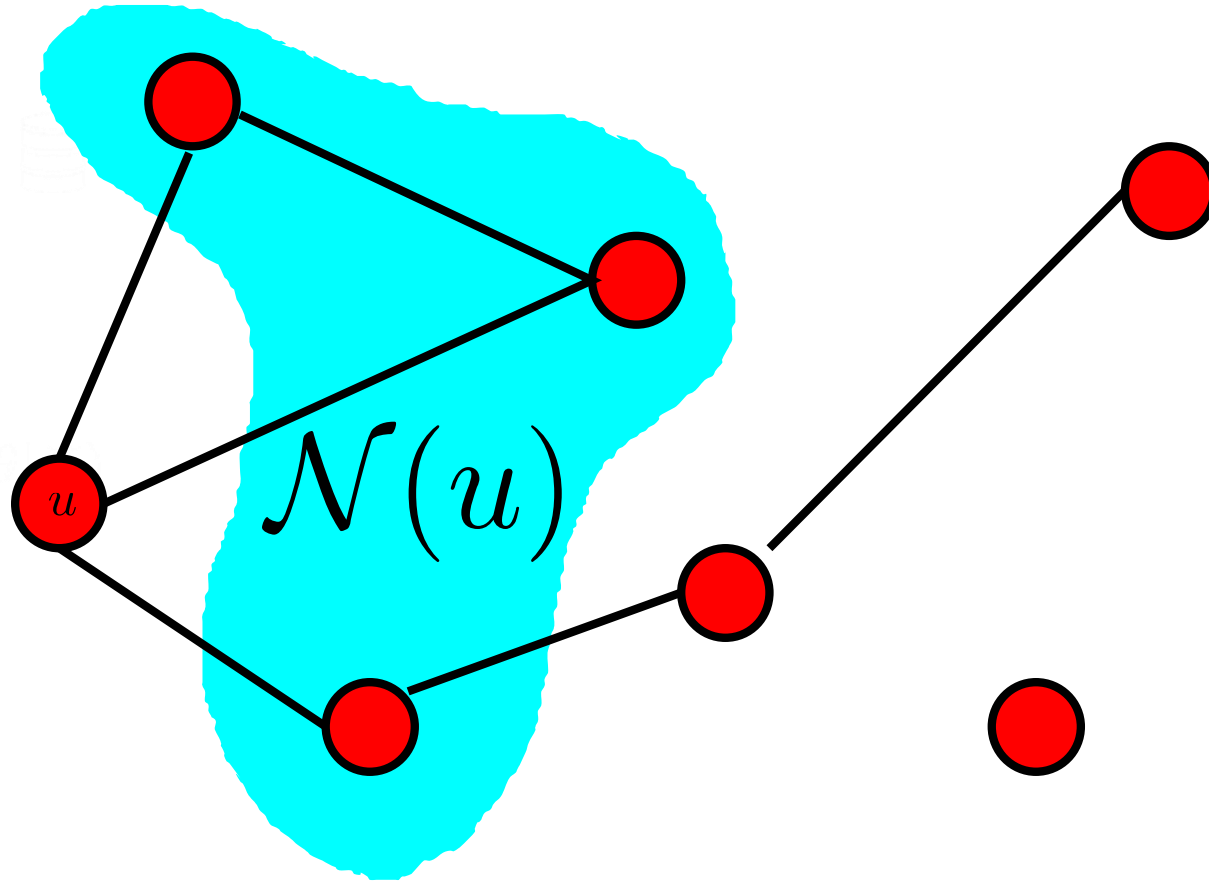
Definition (undirected graph)**.** $G = (V, E)$ *is* ***undirected*** *if all edges go in both directions: for all* $(u, v) \in V \times V$ *such that* $u \neq v$*, we have:*

$$(u, v) \in E \iff (v, u) \in E.$$



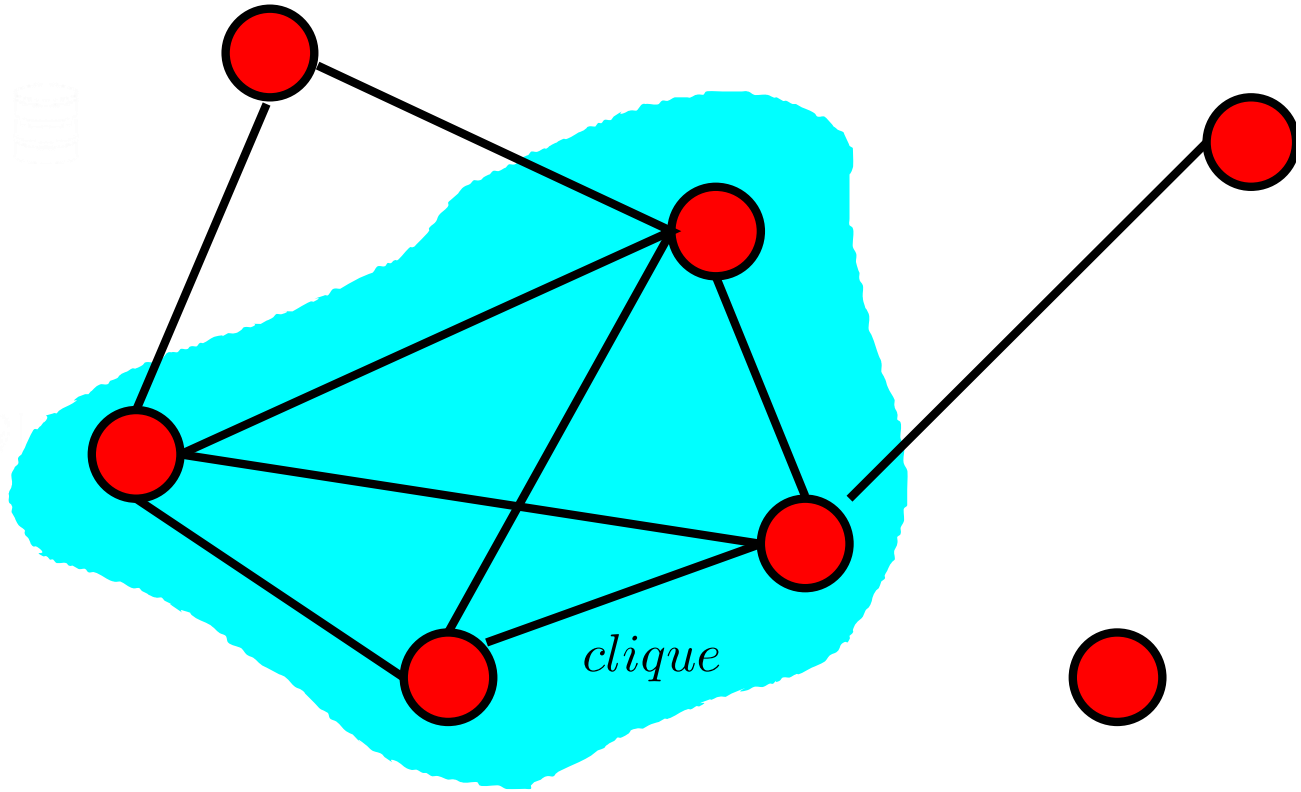
What's a neighbourhood?

Definition. *The set of* ***neighbours*** *of* u*, is* $\mathcal{N}(u) = \{v \in V, (v, u) \in E\}$.



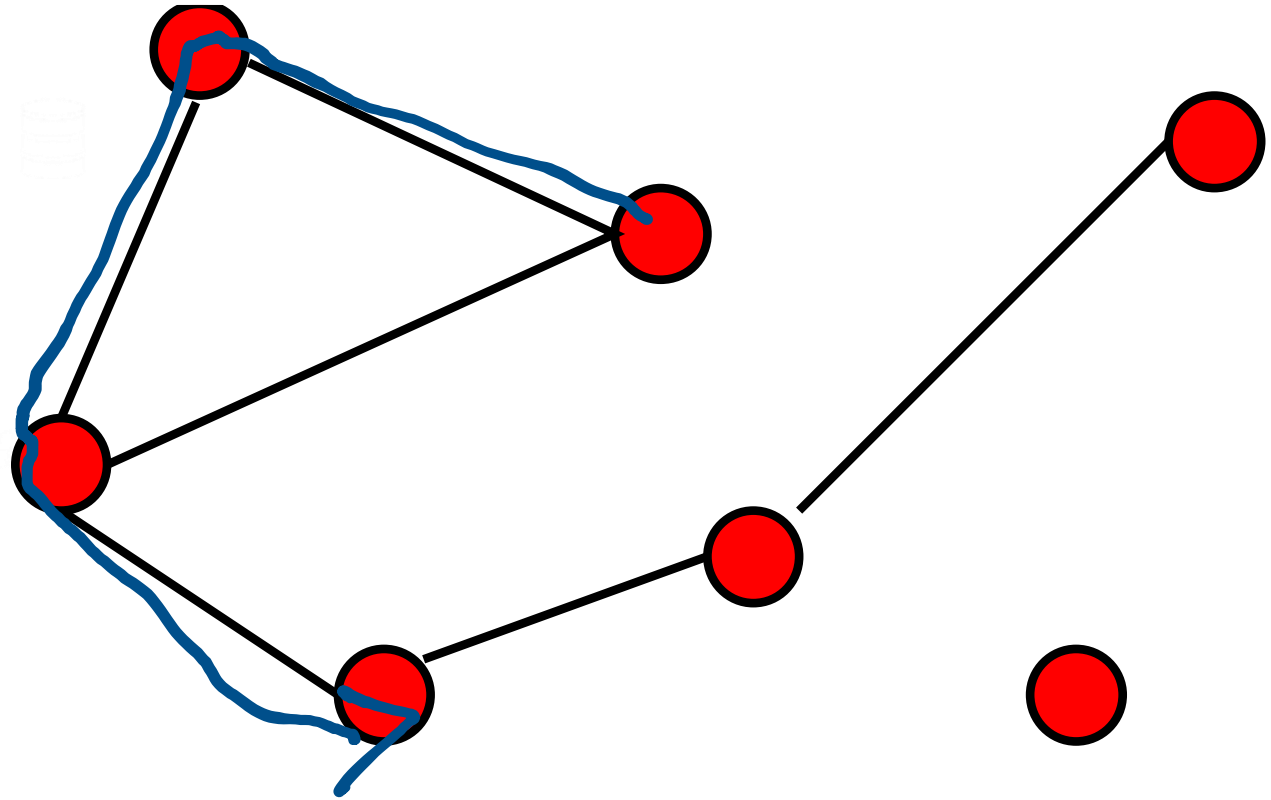
What's a clique?

Definition. *A totally connected subset of vertices is called a* ***clique****.*



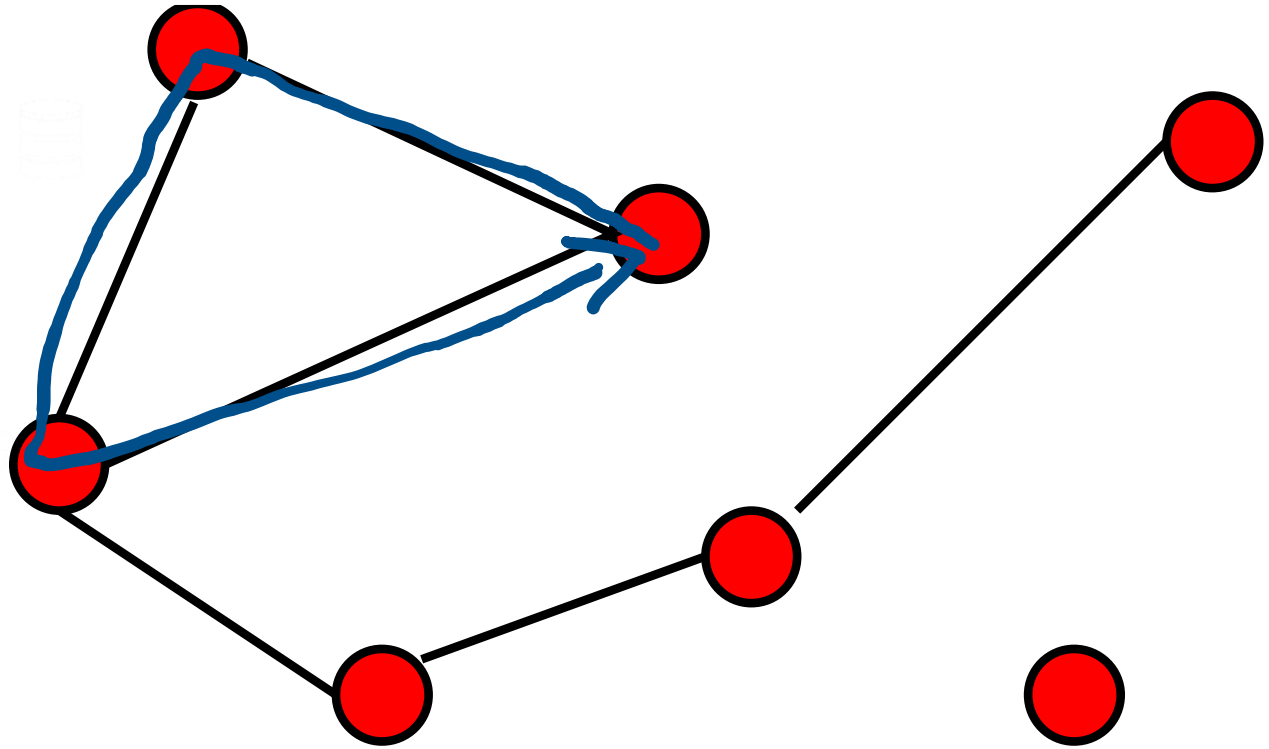
What's a path?

Definition. *A* ***path*** *is a sequence of connected vertices that are globally distinct.*



What's a cycle?

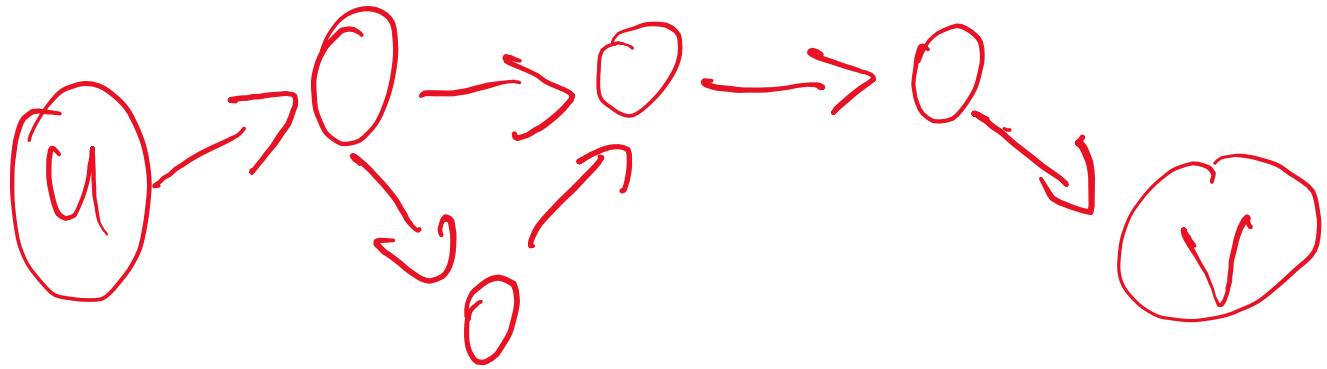
Definition. *A path that begins and end at the same point is called a* ***cycle****.*



Some definitions specific to directed graphs

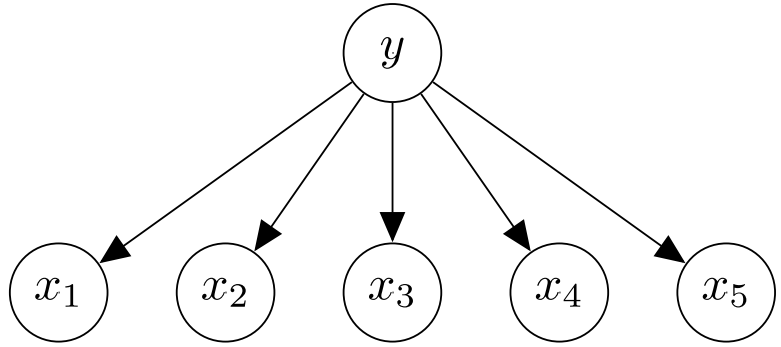
Definition. *u is a **parent** of v if $(u, v) \in E$. We also say that v is a **children** of u.*

Definition. *u is an **ancestor** of v if there exists a path from u to v. We also say that v is a **descendant** of u.*

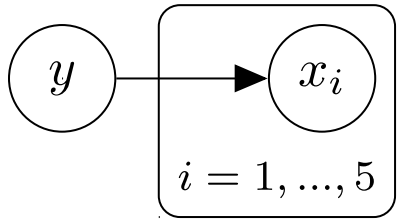


The useful plate notation

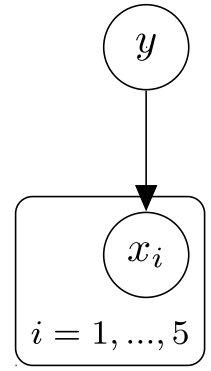
- Often we deal with graphs with recurring stucture, for instance



- The **plate notation** is a more compact way of drawing the same graph:



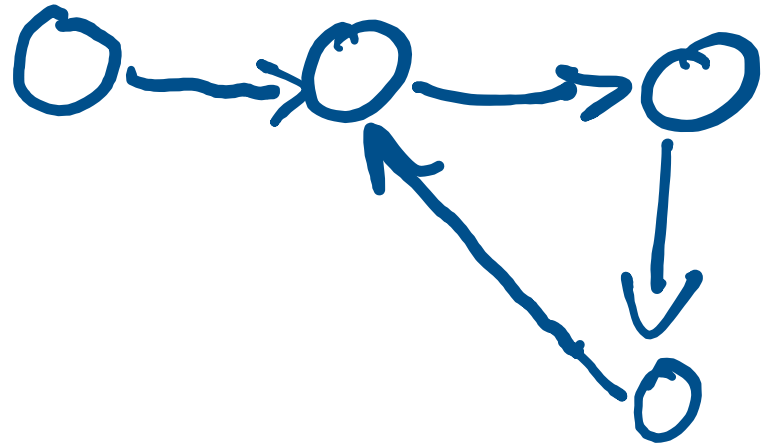
or



A key concept: DAGs

Definition (DAG). *A **directed acyclic graph** (DAG) is a directed graph without any cycle.*

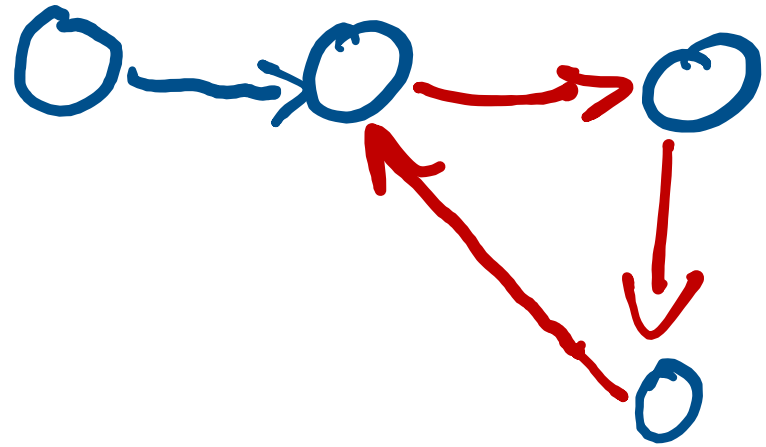
- Is this a DAG?



A key concept: DAGs

Definition (DAG). *A **directed acyclic graph** (DAG) is a directed graph without any cycle.*

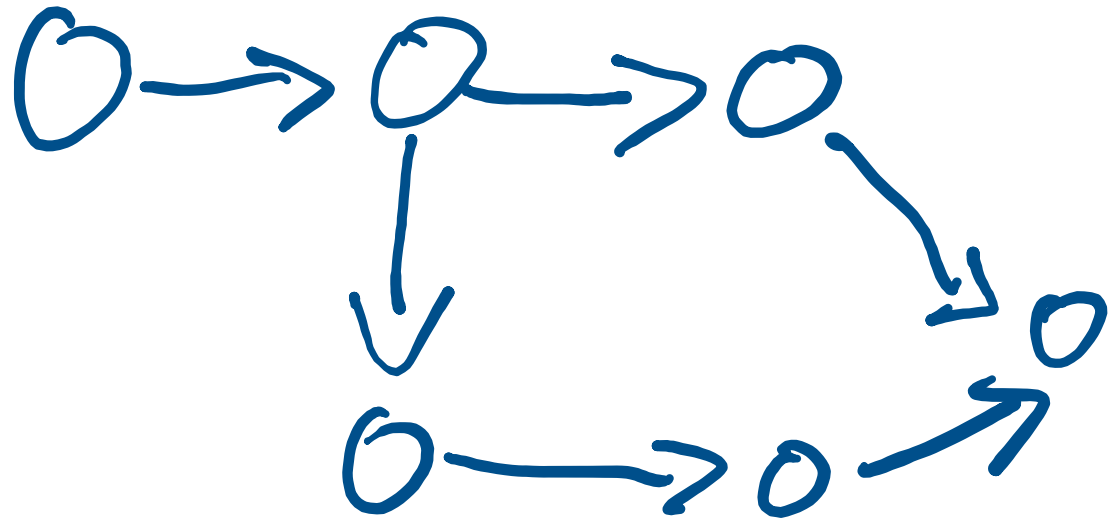
- Is this a DAG?
- **No!** There is a cycle.



A key concept: DAGs

Definition (DAG). *A **directed acyclic graph** (DAG) is a directed graph without any cycle.*

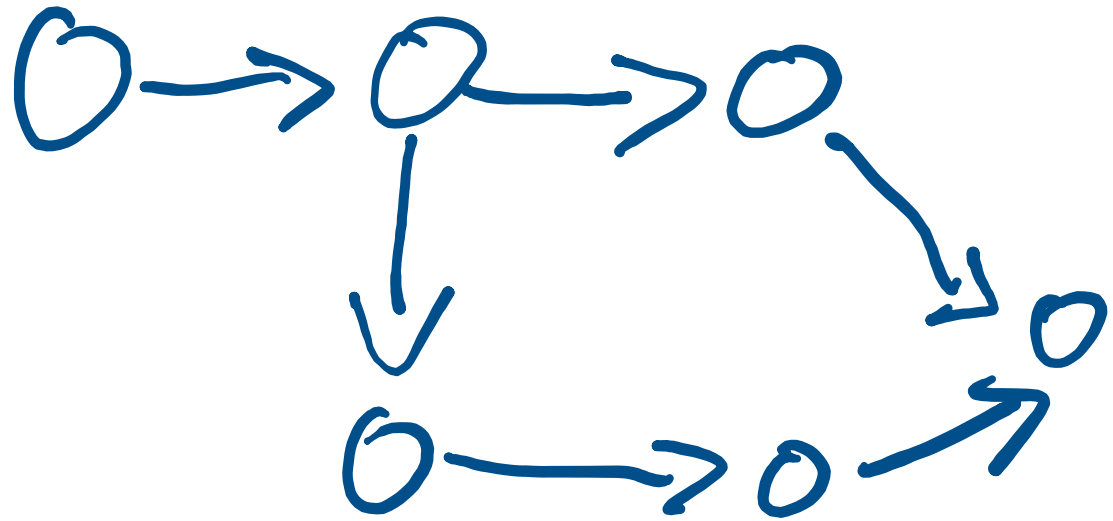
- Is this a DAG?



A key concept: DAGs

Definition (DAG). *A **directed acyclic graph** (DAG) is a directed graph without any cycle.*

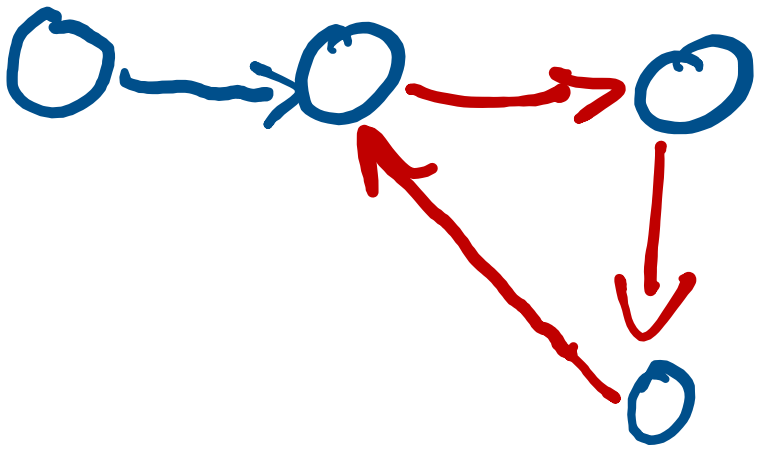
- Is this a DAG?
- **Yes!** No cycle here.



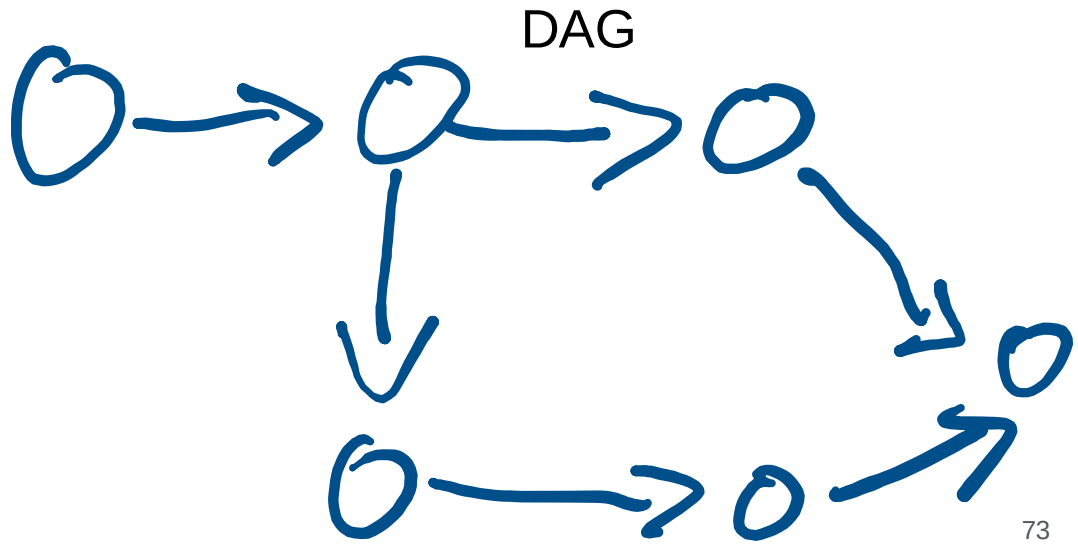
A key concept: DAGs

- All directed graphs we'll look at will be DAGs, so it's a **really important notion!**
- It's also super important for other things than graphical models. For instance, in deep learning, **backpropagation** usually requires that your neural net is a DAG.

Not a DAG



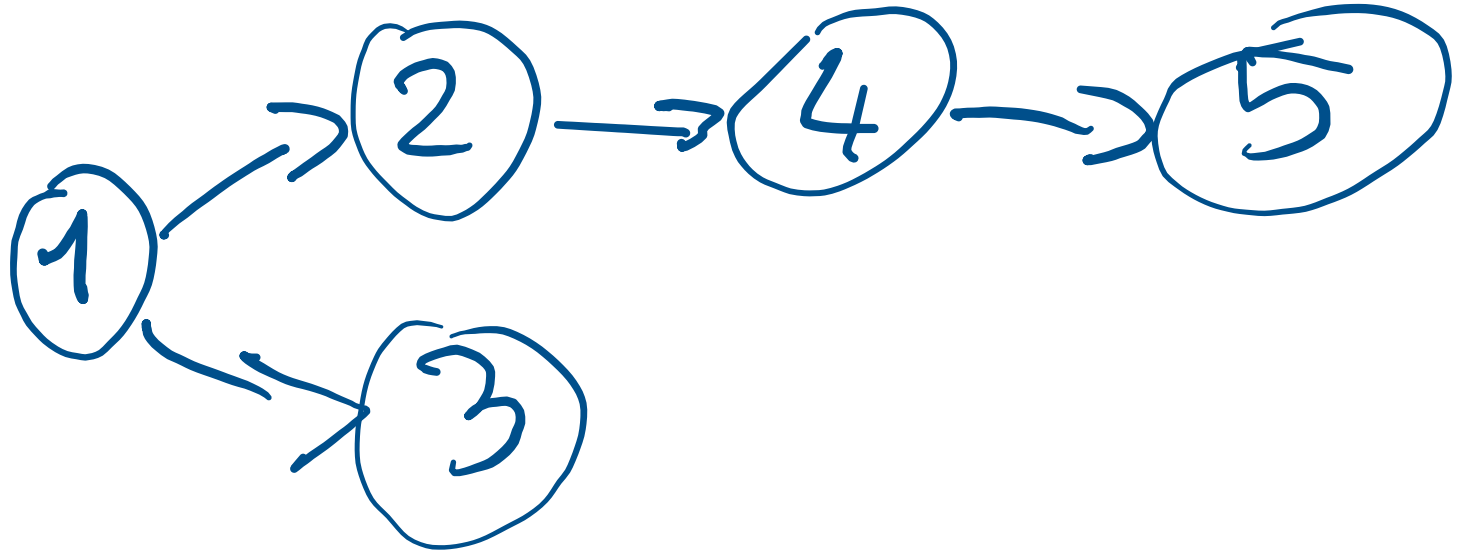
DAG



Why are DAGs very useful?

- One of the most compelling properties of DAGs is that it is possible to order their nodes such that **any parent has a smaller order than all its descendants**. Such an ordering is called a **topological ordering**. The order of a node v is denoted by $I(v)$.

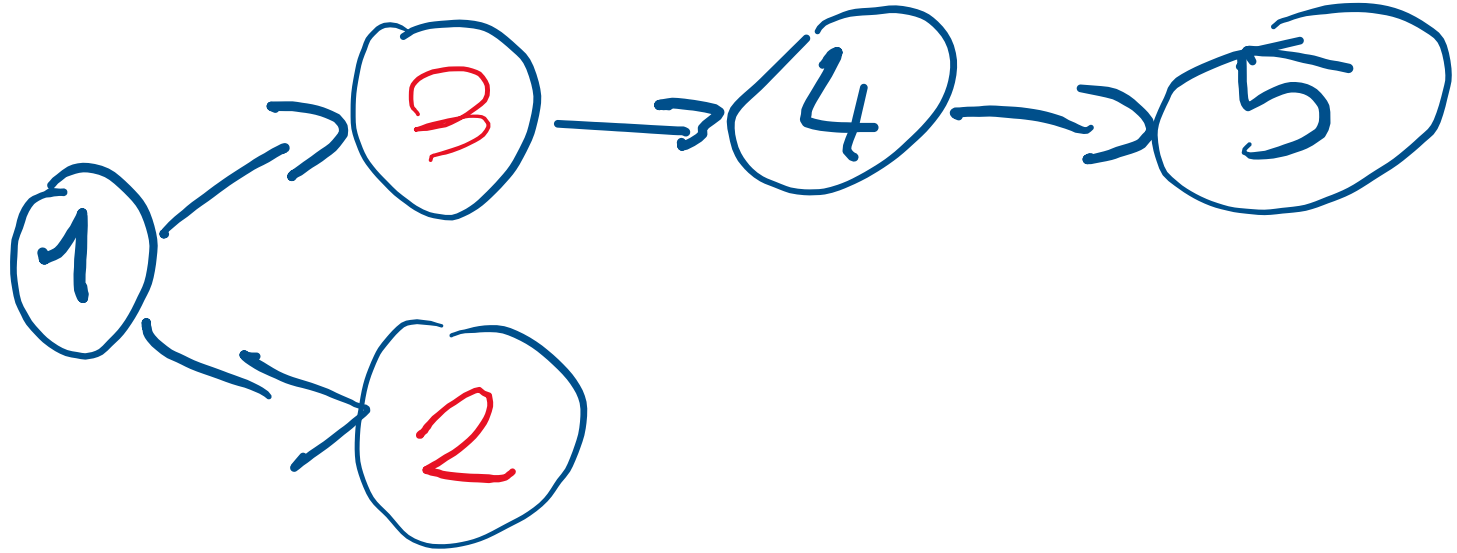
A topologically ordered DAG



Why are DAGs very useful?

- One of the most compelling properties of DAGs is that it is possible to order their nodes such that **any parent has a smaller order than all its descendants**. Such an ordering is called a **topological ordering**. The order of a node v is denoted by $I(v)$.

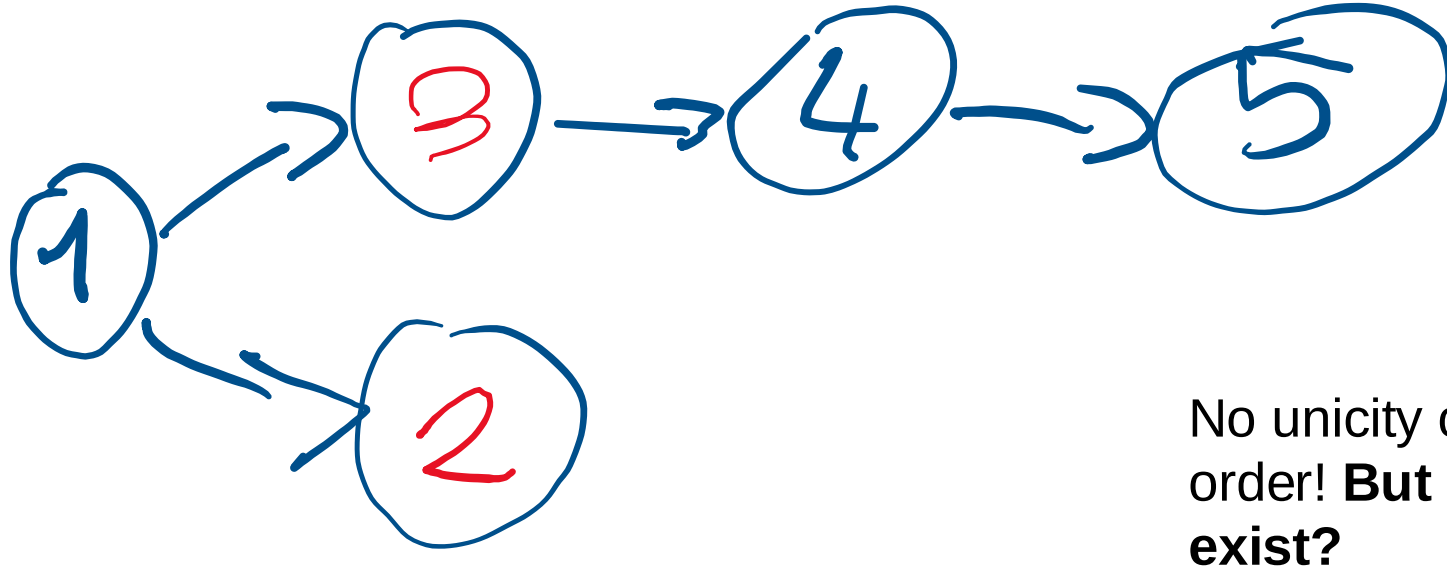
The same DAG with a different topological order



Why are DAGs very useful?

- One of the most compelling properties of DAGs is that it is possible to order their nodes such that **any parent has a smaller order than all its descendants**. Such an ordering is called a **topological ordering**. The order of a node v is denoted by $I(v)$.

The same DAG with a different topological order



No unicity of topological order! **But do they always exist?**

The topological ordering characterises DAGs!

Theorem. *A directed graph can be topologically ordered if, and only if, it is a DAG.*

Proof.

The topological ordering characterises DAGs!

Theorem. *A directed graph can be topologically ordered if, and only if, it is a DAG.*

Proof. (order $\Longrightarrow$ DAG) By contradiction: if there were a cycle from u to u, then u would be a descendant of u, so the topological properties would imply $I(u) < I(u)$, which is impossible.

The topological ordering characterises DAGs!

Theorem. *A directed graph can be topologically ordered if, and only if, it is a DAG.*

Proof. (DAG $\Longrightarrow$ order) It is possible to order any DAG with d nodes by following this simple algorithm. The idea is to rank sequentially the graph, starting with the eldest. After each step, all the nodes that we've already ordered are marked so we know what's left to order.

Algorithm 1 Simple DAG ordering

```
Set all nodes unmarked.
for k = 1, ..., d do
    Select any unmarked node u whose parents are all marked.
    I(u) = k
    Mark u.
end for
```

Aparté: Internships at Inria Nice

- We’ll have several internship proposals at Inria Nice
  - ➢ One on missing data with Aude Sportisse and me, I’ll keep you updated!

3

Directed graphical models (aka Bayesian networks)

What are graphical models?

- A graphical model is a **family of probability distributions that can be factorised in a way that is described by a given graph.**
- **Each node of the graph will correspond to a random variable.**
- The definition depends if the graph is directed or not, but the key idea remains the same: we want to **draw edges between variables when they are directly dependent** (this is hand-wavy and somewhat false, but not a too bad way of seeing things).
- Directed graphical models correspond to the most commonly used statistical models, so we will start with them!
- In this section, G **will always be a DAG.**

What are directed graphical models?

- Let $G = (V, E)$ be a DAG whose nodes are denoted $V = \{1, ..., d\}$
- Let $X = (X_1, ..., X_d)$ be a random variable with density $p(x) = p(x_1, ..., x_d)$.
- Every node of the graph corresponds to a random variable (one vertex for each feature)

What are directed graphical models?

- Let $G = (V, E)$ be a DAG whose nodes are denoted $V = \{1, ..., d\}$
- Let $X = (X_1, ..., X_d)$ be a random variable with density $p(x) = p(x_1, ..., x_d)$.
- Every node of the graph corresponds to a random variable (one vertex for each feature)

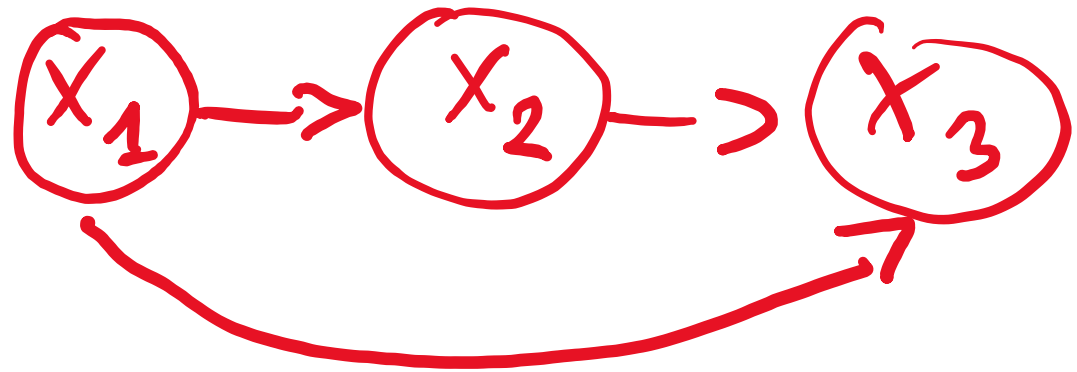
Definition (Directed graphical model)**.** *We say that* p ***factorises in*** G *(denoted* $p \in \mathcal{L}(G)$*) when, for all* x*,*

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i}),$$

where, for all node i*,* pa_i *denotes the set of parents of node* i*.*

First examples of directed graphical models: chain rule

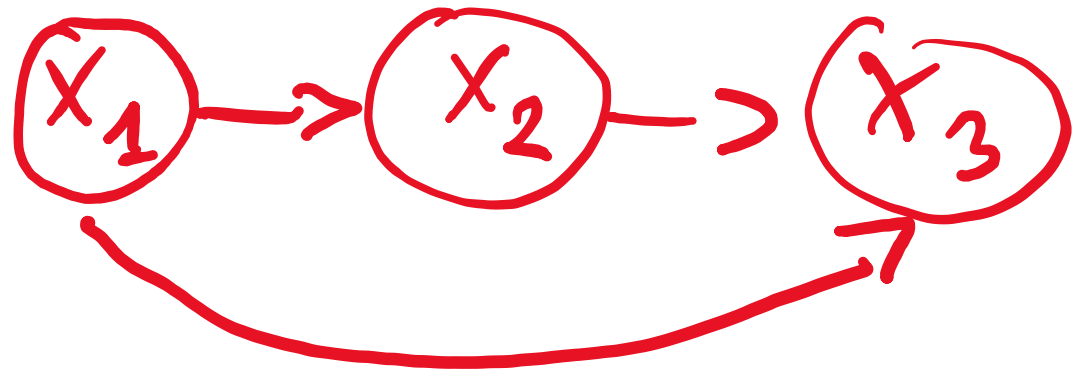
- You already know some models that satisfy $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$, for instance
- The chain rule $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$ for $d = 3$, implies that **any** $p(x)$ factorises as the graph



- **Can you see why?**

First examples of directed graphical models: chain rule

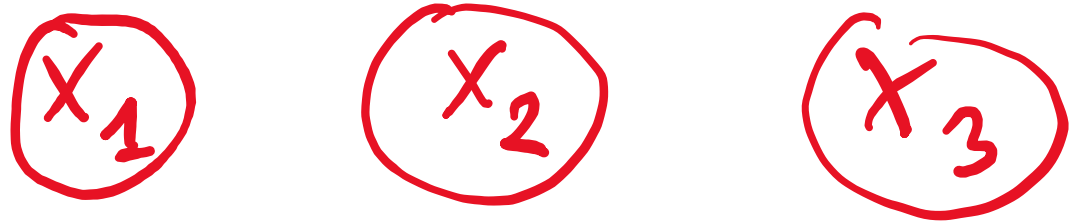
- You already know some models that satisfy $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$, for instance
- The chain rule $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)$, for $d = 3$, implies that **any** $p(x)$ factorises as the graph



- Indeed, x_1 has no parent, x_2 is the parent of x_3, and x_1, x_2 are the parents of x_3.

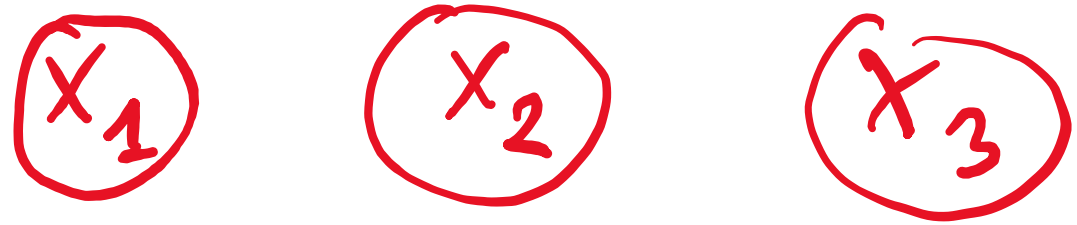
First examples of directed graphical models: independence

- You already know some models that satisfy $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$, for instance
- When all $d = 3$ variables are independent, then $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3)$, and the distribution factorises as



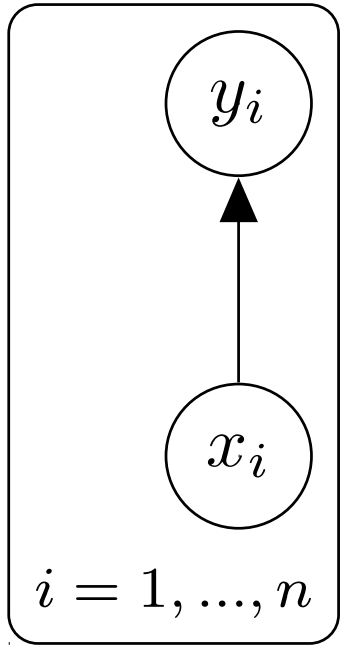
First examples of directed graphical models: independence

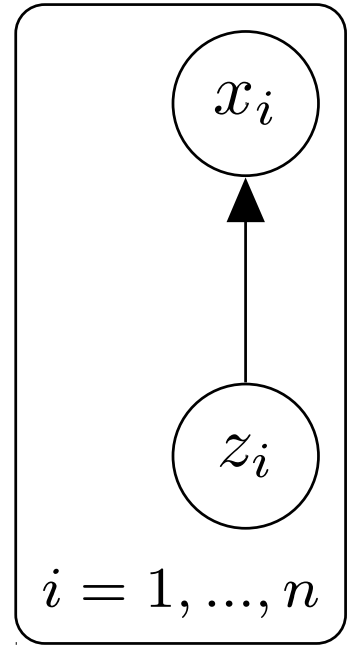
- You already know some models that satisfy $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$, for instance
- When all $d = 3$ variables are independent, then $p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3)$, and the distribution factorises as



- Indeed, no node has any parent.

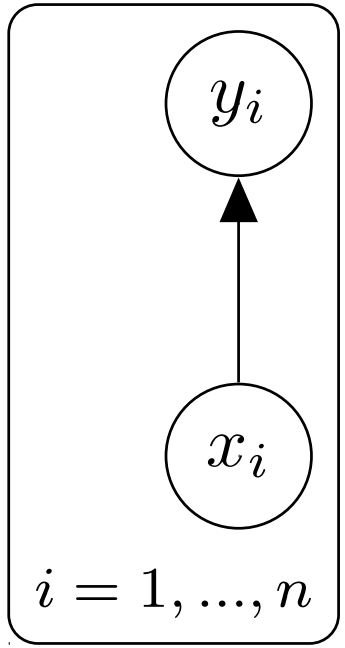
First examples of directed graphical models: linear regression and GMM

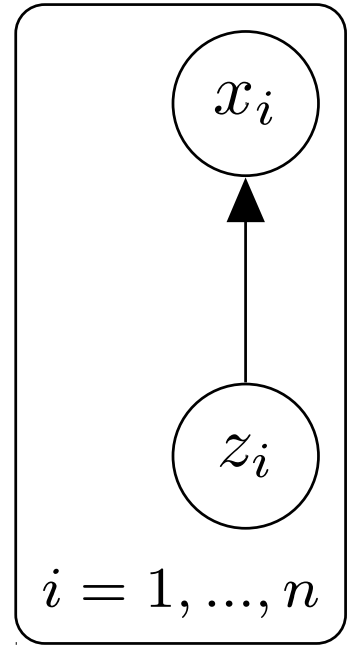




- They look the same! **Q:** Why are they different?

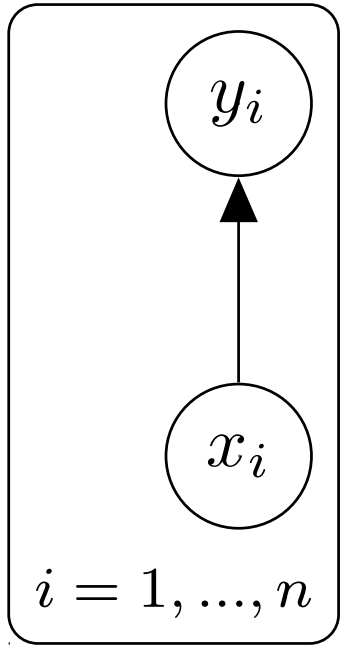
First examples of directed graphical models: linear regression and GMM

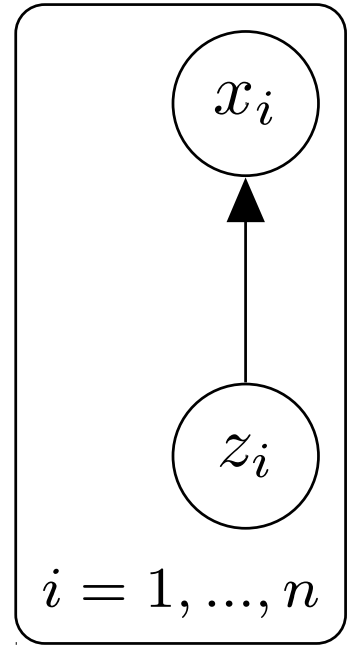




- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z

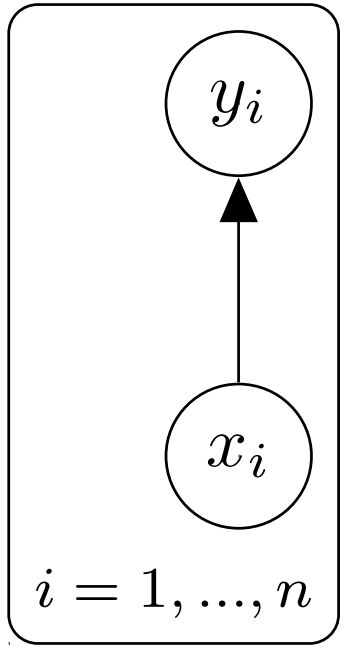
First examples of directed graphical models: linear regression and GMM

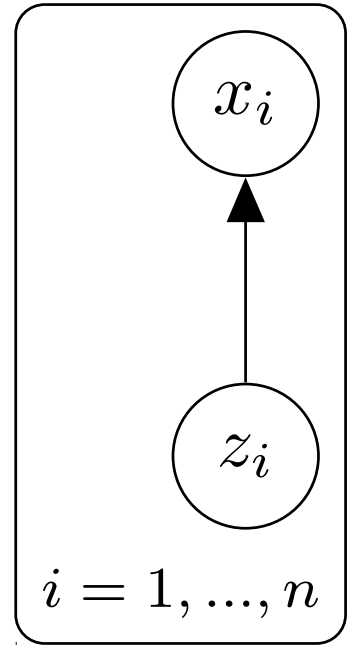




- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z
    2. They are also parametrised differently

First examples of directed graphical models: linear regression and GMM

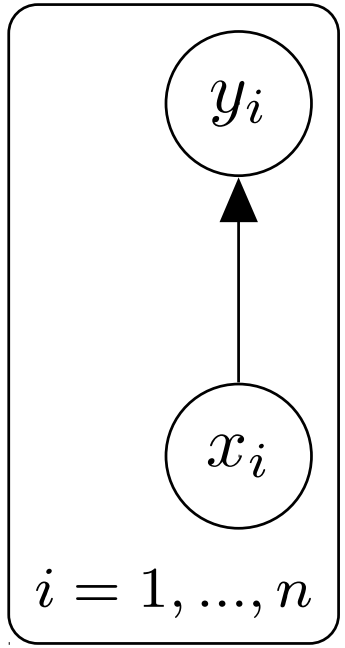


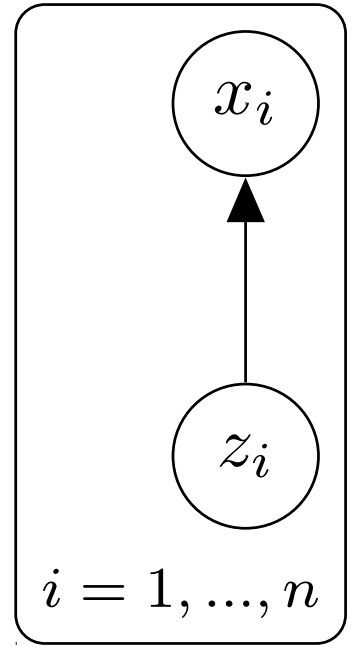


$$p_{\beta,\sigma}(y_i|x_i) = \mathcal{N}(y_i|x_i^T\beta, \sigma^2)$$

- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z
    2. They are also parametrised differently

First examples of directed graphical models: linear regression and GMM



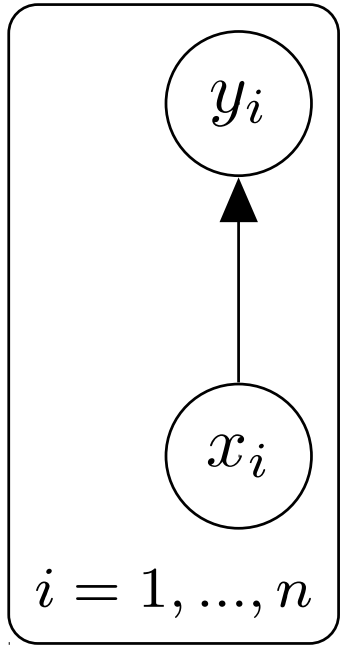


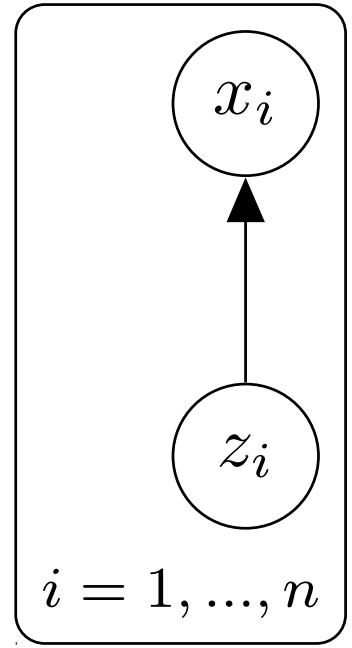
$$p_{\beta,\sigma}(y_i|x_i) = \mathcal{N}(y_i|x_i^T\beta, \sigma^2)$$

$$p_{\boldsymbol{\mu},\boldsymbol{\sigma}}(x_i|z_i) = \mathcal{N}(x_i|\mu_{z_i}, \sigma^2_{z_i})$$

- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z
    2. They are also parametrised differently

First examples of directed graphical models: linear regression and GMM



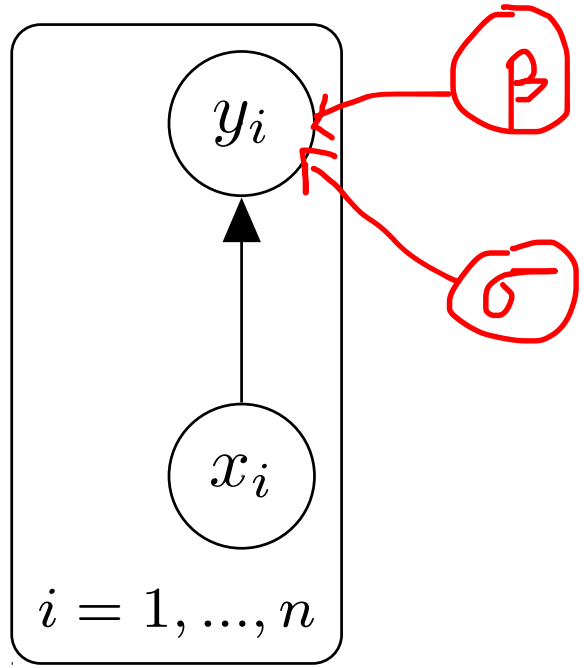


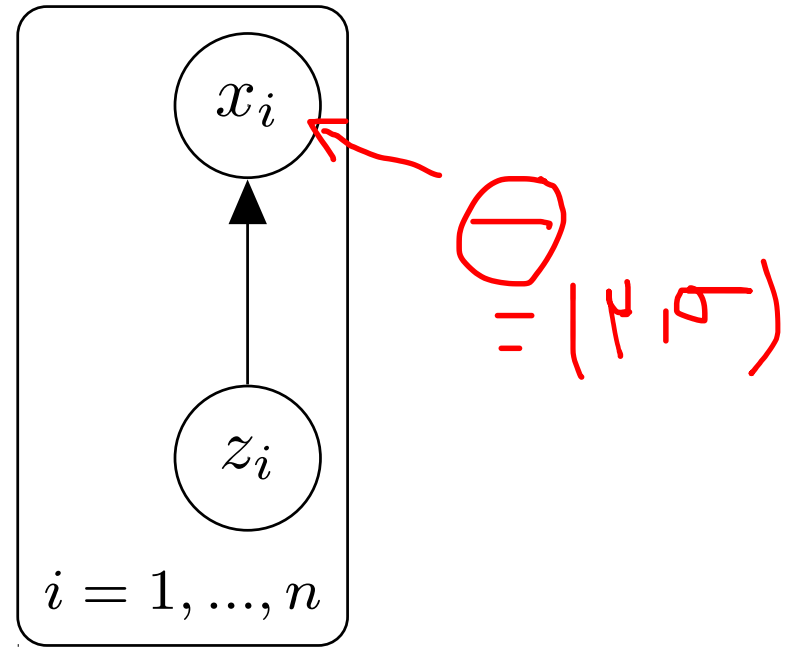
$$p_{\beta,\sigma}(y_i|x_i) = \mathcal{N}(y_i|x_i^T\beta, \sigma^2)$$

$$p_{\boldsymbol{\mu},\boldsymbol{\sigma}}(x_i|z_i) = \mathcal{N}(x_i|\mu_{z_i}, \sigma^2_{z_i})$$

- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z
    2. They are also parametrised differently
- **Q:** Could we use a similar model for our beloved binary MNIST?

First examples of directed graphical models: linear regression and GMM





$$p_{\beta,\sigma}(y_i|x_i) = \mathcal{N}(y_i|x_i^T\beta, \sigma^2)$$

$$p_{\boldsymbol{\mu},\boldsymbol{\sigma}}(x_i|z_i) = \mathcal{N}(x_i|\mu_{z_i}, \sigma^2_{z_i})$$

- They look the same! **Q:** Why are they different?
    1. In linear regression, we observe both x and y, in GMMs we observe only x, and not z
    2. They are also parametrised differently
- **Q:** Could we use a similar model for our beloved binary MNIST? e.g. $p_{\boldsymbol{\theta}}(x_i|z_i) = \mathcal{B}(x_i|\theta_{z_i})$

Let’s go back to the definition…

- A graphical model $\mathcal{L}(G)$ is a family of probability distributions that can be factorised as

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$$

- In maths, when we have factorisation results, a natural question is: **is the factorisation unique?**

Let’s go back to the definition…

- A graphical model $\mathcal{L}(G)$ is a family of probability distributions that can be factorised as

$$p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$$

- In maths, when we have factorisation results, a natural question is: **is the factorisation unique?**

- For instance, a classical arithmetics theorems asserts that **any natural number can be written as a product of prime numbers in a unique way**.

- Here, do you think that a decomposition like $p(x) = \prod_{i=1}^{d} p(x_i | x_{\mathrm{pa}_i})$, when it exists, is unique in a sense?

DAG factorisations are unique

Theorem (unicity of the DAG factorisation). *Let $p(x)$ be a distribution such that there exists some nonnegative functions $f_1, ..., f_d \geq 0$ such that, for all x,*

$$p(x) = \prod_{i=1}^{d} f_i(x_i, x_{pa_i})$$

and, for all $i \in \{1, ..., d\}$ and for all x_i, x_{pa_i},

$$\int_{x_i} f_i(x_i, x_{pa_i}) dx_i = 1.$$

Then, for all $i \in \{1, ..., d\}$,

$$f_i(x_i, x_{pa_i}) = p(x_i | x_{pa_i}).$$

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

$\mathbb{P}(\theta|\mathcal{X})$

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We start with the factor $p(x_d|x_{\mathrm{pa}_d})$ that corresponds to this leaf. A first remark is that $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\mathrm{pa}_d})$, indeed

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We start with the factor $p(x_d|x_{\mathrm{pa}_d})$ that corresponds to this leaf. A first remark is that $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\mathrm{pa}_d})$, indeed

$$p(x_d|x_1, ...x_{d-1}) = \frac{p(x_1, ...x_{d-1}, x_d)}{p(x_1, ...x_{d-1})} = f_d(x_d, x_{\mathrm{pa}_d}) \frac{\prod_{i=1}^{d-1} f_i(x_i, x_{\mathrm{pa}_i})}{p(x_1, ...x_{d-1})},$$

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We start with the factor $p(x_d|x_{\mathrm{pa}_d})$ that corresponds to this leaf. A first remark is that $p(x_d|x_1,...x_{d-1}) = f_d(x_d, x_{\mathrm{pa}_d})$, indeed

$$p(x_d|x_1,...x_{d-1}) = \frac{p(x_1,...x_{d-1},x_d)}{p(x_1,...x_{d-1})} = f_d(x_d, x_{\mathrm{pa}_d}) \frac{\prod_{i=1}^{d-1} f_i(x_i, x_{\mathrm{pa}_i})}{p(x_1,...x_{d-1})},$$

does not depend on x_d because x_d is the parent of no-one!

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We start with the factor $p(x_d|x_{\mathrm{pa}_d})$ that corresponds to this leaf. A first remark is that $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\mathrm{pa}_d})$, indeed

$$p(x_d|x_1, ...x_{d-1}) = \frac{p(x_1, ...x_{d-1}, x_d)}{p(x_1, ...x_{d-1})} = f_d(x_d, x_{\mathrm{pa}_d}) \frac{\prod_{i=1}^{d-1} f_i(x_i, x_{\mathrm{pa}_i})}{p(x_1, ...x_{d-1})},$$

does not depend on x_d because x_d is the parent of no-one!

which means that $p(x_d|x_1, ...x_{d-1}) \propto f_d(x_d, x_{\mathrm{pa}_d})$, and since both $p(x_d|x_1, ...x_{d-1})$ and $f_d(x_d, x_{\mathrm{pa}_d})$ are density functions, we have $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\mathrm{pa}_d})$.

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We start with the factor $p(x_d|x_{\text{pa}_d})$ that corresponds to this leaf. A first remark is that $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\text{pa}_d})$, indeed

$$p(x_d|x_1, ...x_{d-1}) = \frac{p(x_1, ...x_{d-1}, x_d)}{p(x_1, ...x_{d-1})} = f_d(x_d, x_{\text{pa}_d}) \frac{\prod_{i=1}^{d-1} f_i(x_i, x_{\text{pa}_i})}{p(x_1, ...x_{d-1})},$$

does not depend on x_d because x_d is the parent of no-one!

which means that $p(x_d|x_1, ...x_{d-1}) \propto f_d(x_d, x_{\text{pa}_d})$, and since both $p(x_d|x_1, ...x_{d-1})$ and $f_d(x_d, x_{\text{pa}_d})$ are density functions, we have $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\text{pa}_d})$.

Now, since $p(x_d|x_1, ...x_{d-1}) = f_d(x_d, x_{\text{pa}_d})$, this means that $p(x_d|x_1, ...x_{d-1})$ is just a function of x_d and x_{pa_d}, therefore $p(x_d|x_1, ...x_{d-1}) = p(x_d|x_{\text{pa}_d})$ and we finally get

$$f_d(x_d, x_{\text{pa}_d}) = p(x_d|x_{\text{pa}_d}).$$

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We started with the factor $p(x_d|x_{\mathrm{pa}_d})$ and showed that $f_d(x_d, x_{\mathrm{pa}_d}) = p(x_d|x_{\mathrm{pa}_d})$. This initialises our proof by induction!

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We started with the factor $p(x_d|x_{\mathrm{pa}_d})$ and showed that $f_d(x_d, x_{\mathrm{pa}_d}) = p(x_d|x_{\mathrm{pa}_d})$. This initialises our proof by induction!
Now, we will remove the leaf x_d from the graph, leaving us with the distribution $p(x_1, ..., x_{d-1})$. Looking at a ratio from the previous slide, it is clear that

$$p(x_1, ..., x_{d-1}) = \prod_{i=1}^{d-1} f_i(x_i, x_{\mathrm{pa}_i}),$$

so we can redo what we did in the previous slide to show that

$$p(x_{d-1}|x_{\mathrm{pa}_{d-1}}) = f(x_{d-1}|x_{\mathrm{pa}_{d-1}}).$$

DAG factorisations are unique

Proof. We assume, without loss of generality, that the nodes are sorted in topological order. Therefore, the last node x_d has no descendant (it's what we call a **leaf**). The idea will be, starting from this node, to do a proof by induction.

We started with the factor $p(x_d|x_{\mathrm{pa}_d})$ and showed that $f_d(x_d, x_{\mathrm{pa}_d}) = p(x_d|x_{\mathrm{pa}_d})$. This initialises our proof by induction!
Now, we will remove the leaf x_d from the graph, leaving us with the distribution $p(x_1, ..., x_{d-1})$. Looking at a ratio from the previous slide, it is clear that

$$p(x_1, ..., x_{d-1}) = \prod_{i=1}^{d-1} f_i(x_i, x_{\mathrm{pa}_i}),$$

so we can redo what we did in the previous slide to show that

$$p(x_{d-1}|x_{\mathrm{pa}_{d-1}}) = f(x_{d-1}|x_{\mathrm{pa}_{d-1}}).$$

And we can repeat that inductively until we have seen the whole graph, proving finally that $p(x_i|x_{\mathrm{pa}_i}) = f(x_i|x_{\mathrm{pa}_i})$, for all $i \in \{1, ..., d\}$. □

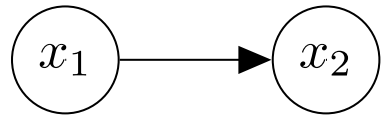
Let’s look at all the possible graphs with 2 nodes

- There are only two possible graphs with two nodes!
- The **trivial graph**, for which $p \in \mathcal{L}(G)$ if, and only if the two variables are independent

Let’s look at all the possible graphs with 2 nodes

- There are only two possible graphs with two nodes!
- The **trivial graph**, for which $p \in \mathcal{L}(G)$ if, and only if the two variables are independent

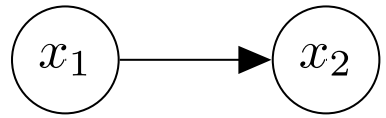
- The **complete graph**, in which any distribution factorises (because of the chain rule)



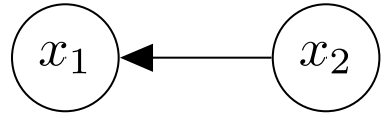
Let’s look at all the possible graphs with 2 nodes

- There are only two possible graphs with two nodes!
- The **trivial graph**, for which $p \in \mathcal{L}(G)$ if, and only if the two variables are independent

- The **complete graph**, in which any distribution factorises (because of the chain rule)



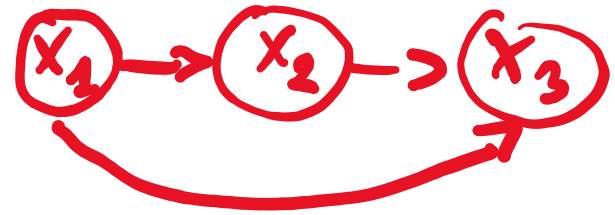
- All complete graphs are equivalent because the chain rule can be used with any ordering of the variables, so the graph above is equivalent to



Let’s look at all the possible graphs with 3 nodes

- It gets a bit trickier! Again, we have the trivial graph that corresponds to independence…

- …and the complete graph in which any distribution factorises:



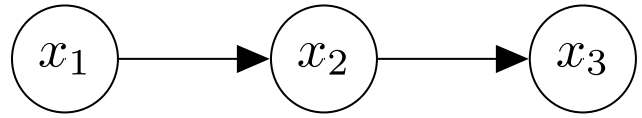
- **But now there are also other graphs!**

Let’s look at all the possible graphs with 3 nodes

- The **Markov chain** corresponds to models with limited memory: if you remember x_2, there is no use in remembering x_1 to predict x_3. This corresponds to the factorisation

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2)$$

and therefore to the graph

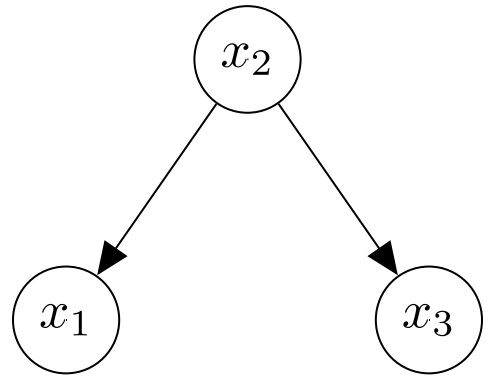


- A consequence is that $x_3 \perp\!\!\!\perp x_1|x_2$.

Let’s look at all the possible graphs with 3 nodes

- The **latent cause** corresponds to the factorisation $p(x) = p(x_2)p(x_1|x_2)p(x_3|x_2)$

and therefore to the graph



- A consequence is that $x_3 \perp\!\!\!\perp x_1|x_2$. That’s the same than for the Markov chain!