Introduction to Probabilistic Models

Antonio Salmerón

Department of Mathematics
University of Almería
antonio.salmeron@ual.es

Trondheim, June 14 2021

Motivation

"I always wondered how it would be if a Superior species landed on Earth and showed us How they played chess. Now I know it."

Peter Heine Nielsen.
Chess Grand Master and Magnus Carlsen's coach

The question is,

- Can we (humans) learn (interpret) anything from it?

Motivation

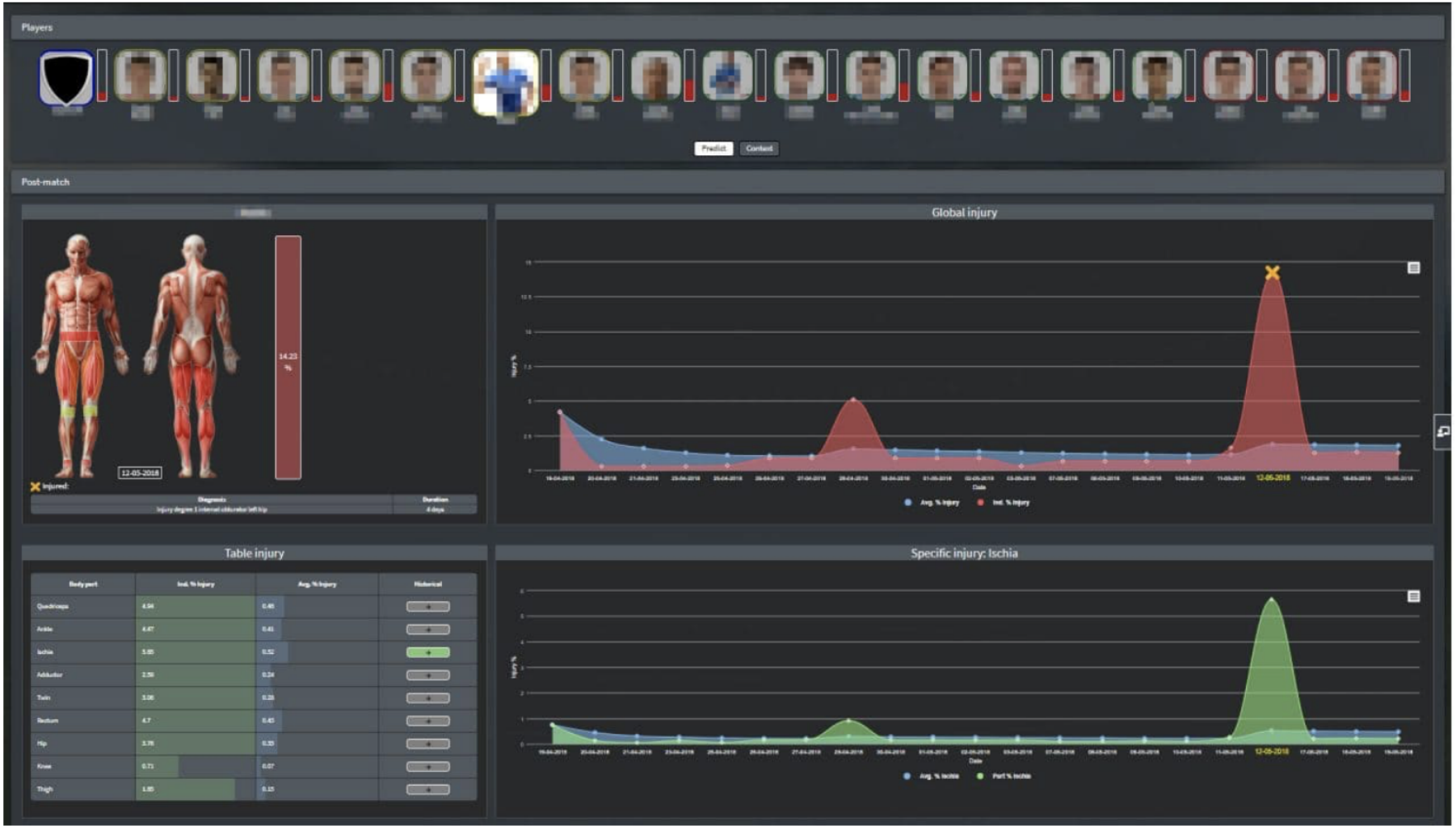
"I always wondered how it would be if a
Superior species landed on Earth and showed us
How they played chess. Now I know it."

Peter Heine Nielsen.
Chess Grand Master and Magnus Carlsen's coach

The question is,

- Can we (humans) learn (interpret) anything from it?

Examples: Predictive medicine



What should I do to prevent players from being injured?

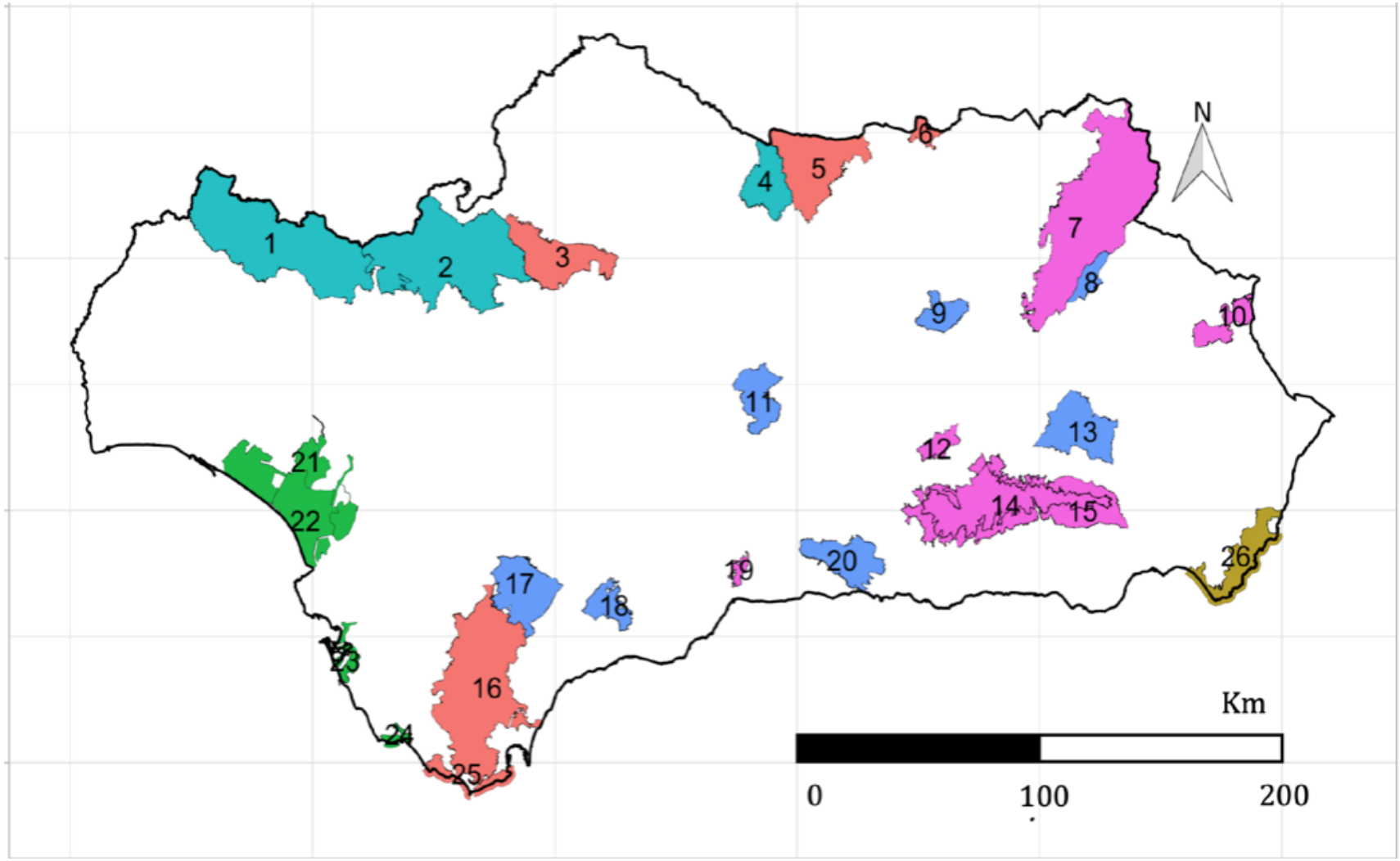
Examples: Self-driving cars

Can I predict an event in advance so that I can avoid it?

Examples: Politics

Designing personalized campaigns?

Examples: Land use



Monitoring protected areas

Probabilistic models

All the previous examples:

- Operate in environments where large amounts of data are available
- However, data don't cover all the possible scenarios
- Use a probabilistic model learnt from data
- Use inference algorithms to carry out prediction and structure analysis

Probabilistic models offer:

- Principled quantification of uncertainty
- Natural way of dealing with missing data
- Interpretability

Probabilistic graphical models

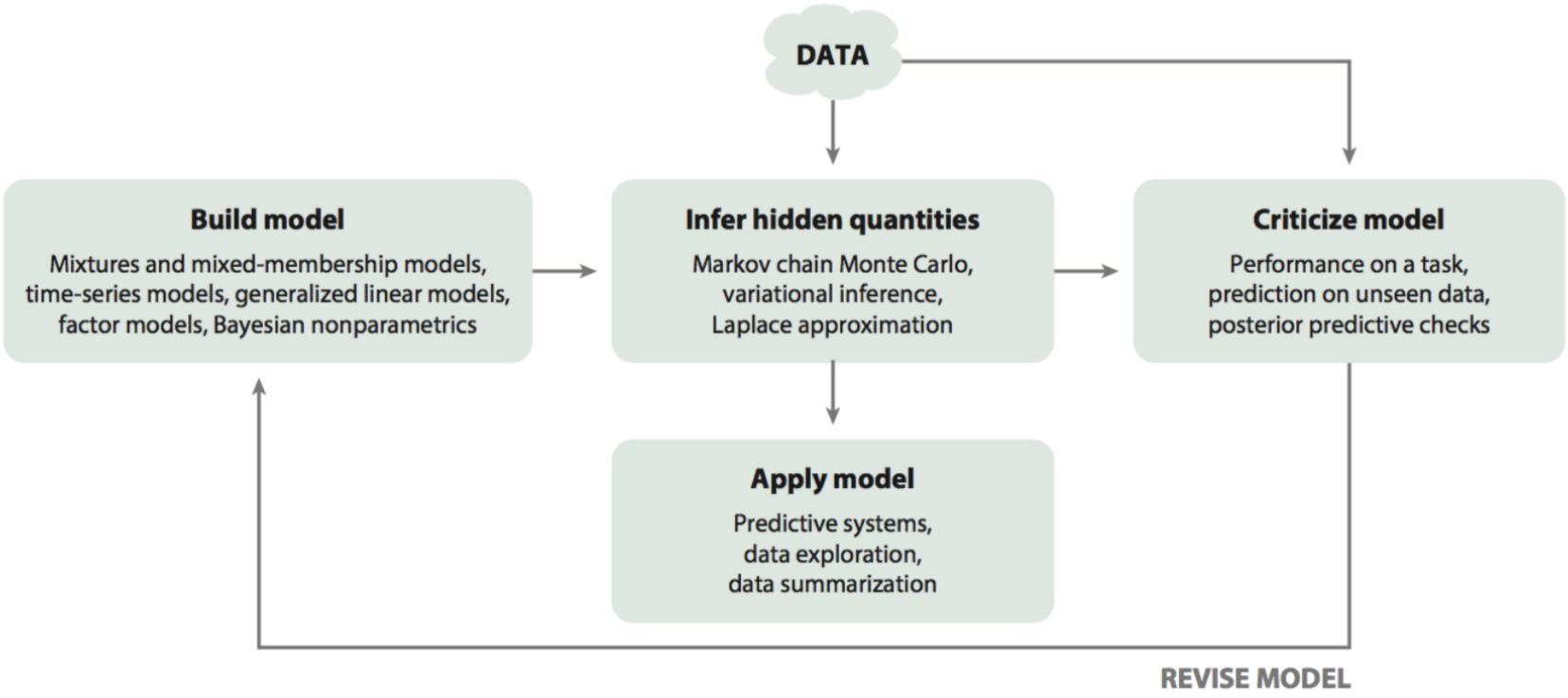
What we need from probabilistic models:

- Ability to operate in high dimensional spaces
- Support efficient inference and learning

Probabilistic graphical models offer:

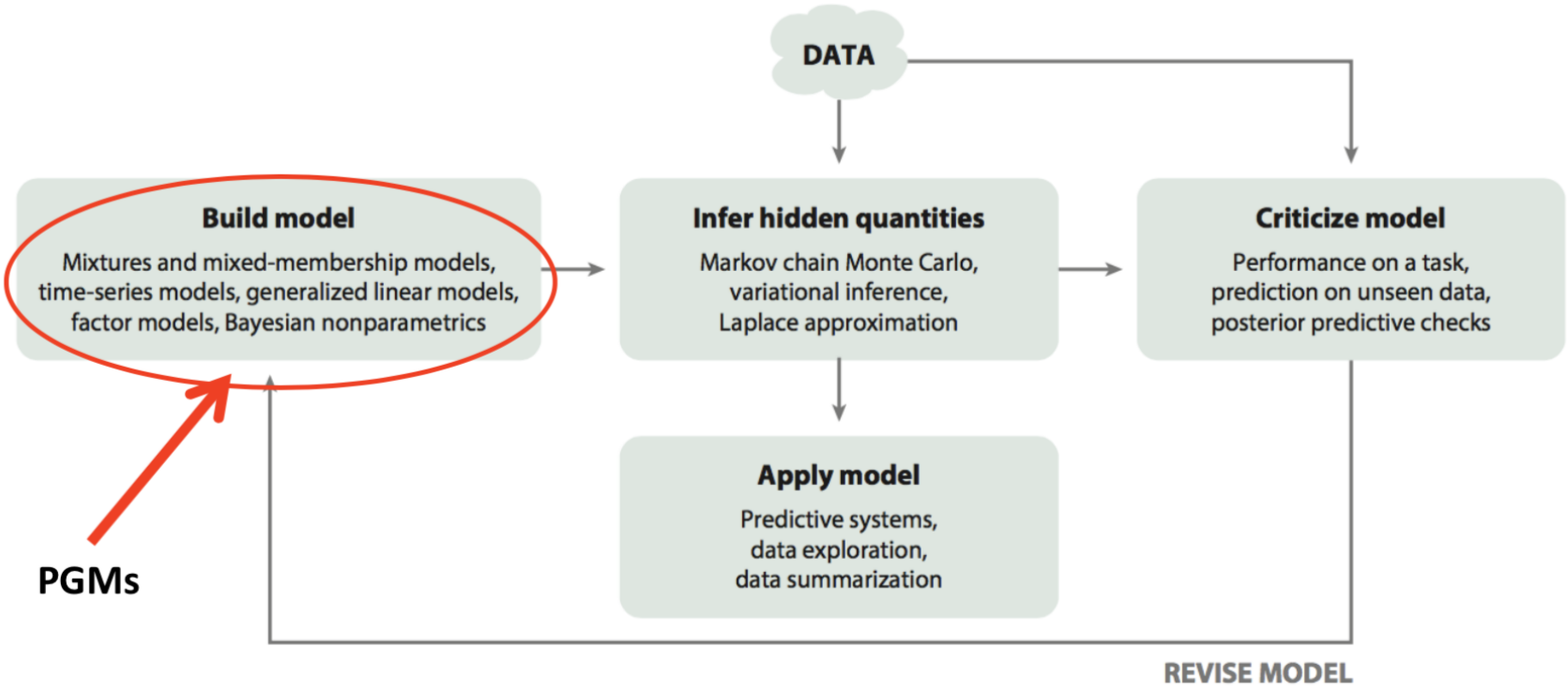
- Structured specification of high dimensional distributions in terms of low dimensional factors
- Efficient inference and learning taking advantage of the structure
- Graphical representation interpretable by humans

The probabilistic modeling cycle



Blei, David M. "Build, compute, critique, repeat: Data analysis with latent variable models." *Annual Review of Statistics and Its Application* 1 (2014): 203-232.

The probabilistic modeling cycle



Blei, David M. "Build, compute, critique, repeat: Data analysis with latent variable models." *Annual Review of Statistics and Its Application* 1 (2014): 203-232.

Learning probabilistic models from data: frequentist approach

- A random variable X
- A probability density/mass function f associated with X
- A parameter θ, whose value is fixed but unknown
- We assume f is known except for parameter θ, and denote this fact as $f(x;\theta)$ or $f(x|\theta)$
- For a sample $X_1, \ldots, X_n$ drawn from $f(x|\theta)$, the likelihood function is

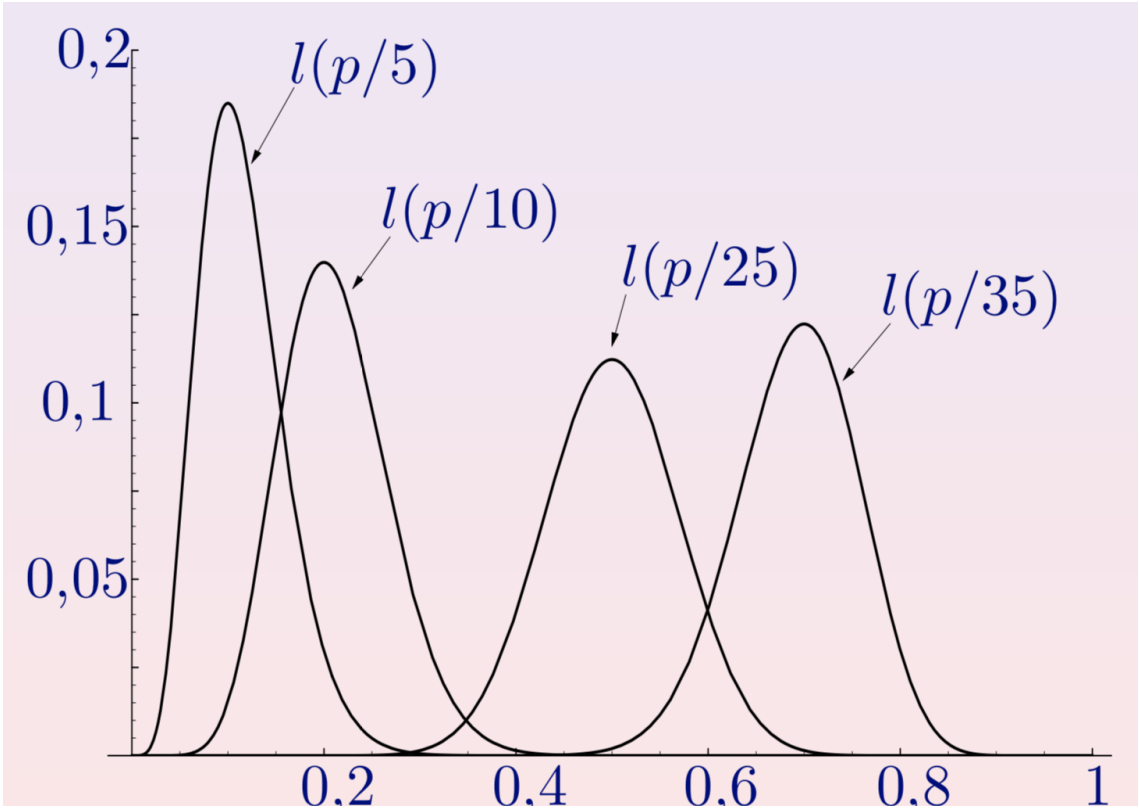
$$l(\theta|x_1, \ldots, x_n) = f(x_1, \ldots, x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

i.e. the joint density/mass of the sample regarded as a function of the parameter

Learning probabilistic models from data: frequentist approach

Assume a sample of size 1, $X \sim \mathcal{B}(50, p)$

The likelihood function is $l(p/x) = \binom{50}{x} p^x (1-p)^{50-x}$



Learning probabilistic models from data: Bayesian approach

- Parameters are modeled as random variables and information about them can be included prior to observing data
- By Bayes' rule, the prior information is combined with the likelihood, yielding a posterior distribution that is used for making inferences about the parameter

Learning probabilistic models from data: Bayesian approach

Distributions in a Bayesian model

- The prior distribution, $\boxed{\pi(\theta)}$
- The joint distribution of (X, θ), $\boxed{\psi(x, \theta) = f(x|\theta)\pi(\theta)}$
- The prior predictive distribution of X, $\boxed{m(x) = \int_\theta f(x|\theta)\pi(\theta)\ d\theta}$
- The posterior distribution of θ,

$$\boxed{\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int_\theta f(x|\theta)\pi(\theta)\ d\theta}}$$

- The predictive distribution given $\boldsymbol{x} = \{x_1, \ldots, x_n\}$:

$$\boxed{f(x_{n+1}|\boldsymbol{x}) = \int_\theta f(x_{n+1}|\theta, \boldsymbol{x})\pi(\theta|\boldsymbol{x})d\theta = \int_\theta f(x_{n+1}|\theta)\pi(\theta|\boldsymbol{x})d\theta}$$

Learning probabilistic models from data: Bayesian approach. Example

- Assume a sample $X_1, X_2, \ldots, X_n \sim \mathcal{B}(1, p)$ and $p \sim \mathcal{U}(0, 1)$
- Then,

$$f(x_1, \ldots, x_n|p) = p^{\sum x_i}(1-p)^{n-\sum x_i}, \quad \text{with } x_i = 0, 1; \quad p \in (0, 1),$$
$$\pi(p) = 1, \text{ if } p \in (0, 1)$$

- The posterior distribution is

$$\pi(p|x_1, \ldots, x_n) = \frac{f(x_1, \ldots, x_n|p)\pi(p)}{\int_0^1 f(x_1, \ldots, x_n|p)\pi(p)\, dp} = \frac{p^{\sum x_i}(1-p)^{n-\sum x_i}}{\int_0^1 p^{\sum x_i}(1-p)^{n-\sum x_i}\, dp}$$
$$= \frac{\Gamma(n+2)}{\Gamma(\sum x_i + 1)\Gamma(n - \sum x_i + 1)} p^{\sum x_i}(1-p)^{n-\sum x_i}$$

which corresponds to $\text{Be}(\sum x_i + 1, n - \sum x_i + 1)$

Very easy to compute for some models

Learning probabilistic models from data: Bayesian approach. Example

- Assume a sample $X_1, X_2, \ldots, X_n \sim \mathcal{B}(1, p)$ and $p \sim \mathcal{U}(0, 1)$
- Then,

$$f(x_1, \ldots, x_n|p) = p^{\sum x_i}(1-p)^{n-\sum x_i}, \quad \text{with } x_i = 0, 1; \quad p \in (0, 1),$$
$$\pi(p) = 1, \text{ if } p \in (0, 1)$$

- The posterior distribution is

$$\pi(p|x_1, \ldots, x_n) = \frac{f(x_1, \ldots, x_n|p)\pi(p)}{\int_0^1 f(x_1, \ldots, x_n|p)\pi(p)\ dp} = \frac{p^{\sum x_i}(1-p)^{n-\sum x_i}}{\int_0^1 p^{\sum x_i}(1-p)^{n-\sum x_i}\ dp}$$
$$= \frac{\Gamma(n+2)}{\Gamma(\sum x_i + 1)\Gamma(n - \sum x_i + 1)} p^{\sum x_i}(1-p)^{n-\sum x_i}$$

which corresponds to $\text{Be}(\sum x_i + 1, n - \sum x_i + 1)$

Very easy to compute for some models

Learning probabilistic models from data: Bayesian approach

- The method above is known as *fully Bayesian* approach
- Sometimes, we don't need to compute the denominator of the posterior distribution, in which case θ can be estimated as

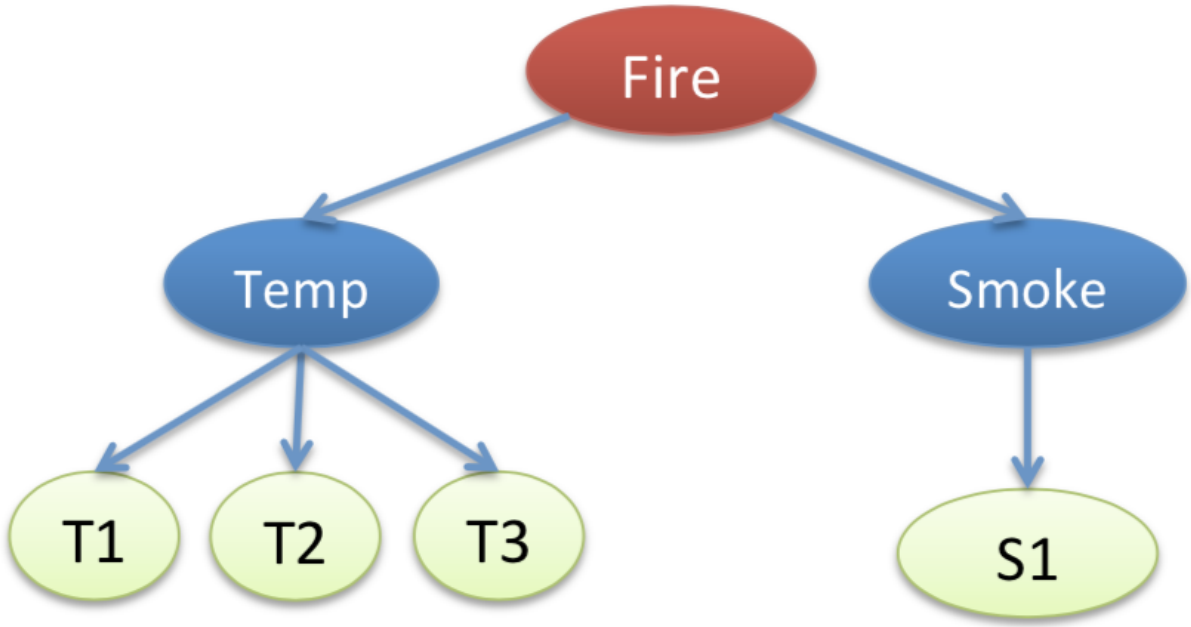
$$\hat{\theta} = \arg\max_{\theta} \log f(x_1, \ldots, x_n|\theta) + \log \pi(\theta)$$

  known as the MAP estimator
- Note that we could also choose

$$\hat{\theta} = \arg\max_{\theta} \log f(x_1, \ldots, x_n|\theta)$$

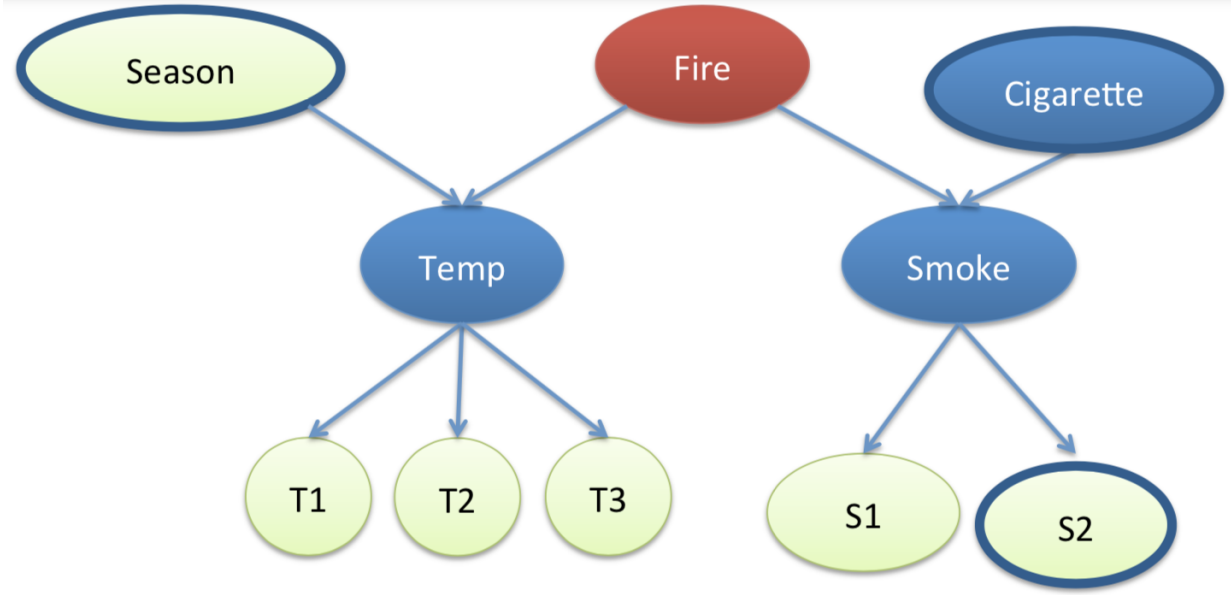
  which is actually the MLE

Bayesian networks



$$p(f, t, s, t_1, t_2, t_3, s_1) = p(t_1|t)p(t_2|t)p(t_3|t)p(s_1|s)p(t|f)p(s|f)p(f)$$

Bayesian networks: modular structure



$$p(se, f, c, t, s, t_1, t_2, t_3, s_1) = p(t_1|t)p(t_2|t)p(t_3|t)p(s_1|s)p(s_2|s)$$
$$p(t|se, f)p(s|f, c)p(se)p(f)p(c)$$

Bayesian networks: definition

A Bayesian network over random variables $X_1, \ldots, X_n$ consists of

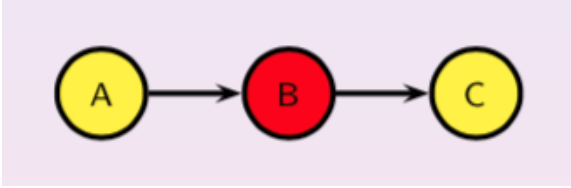
- A DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{X_1, \ldots, X_n\}$
- A set of local conditional distributions $\mathcal{P} = \{p(X_i|pa(X_i)), X_i \in \mathcal{V}\}$ where $pa(X_i)$ denotes the parents of X_i according to $\mathcal{E}$

Every Bayesian network encodes a joint distribution factorized as

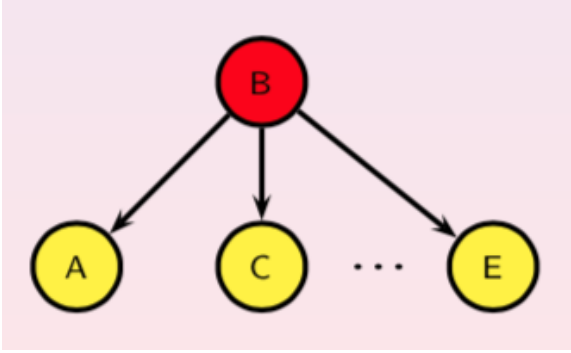
$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|pa(X_i))$$

Interpreting Bayesian network structures: d-separation

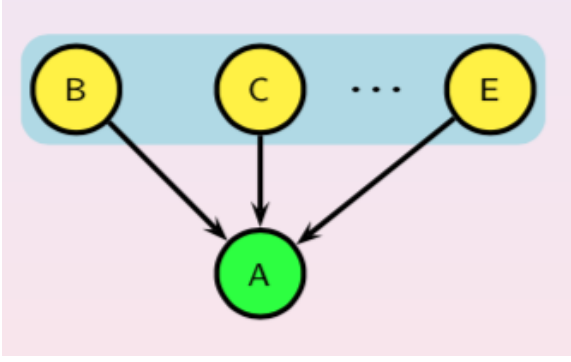
- Serial connection



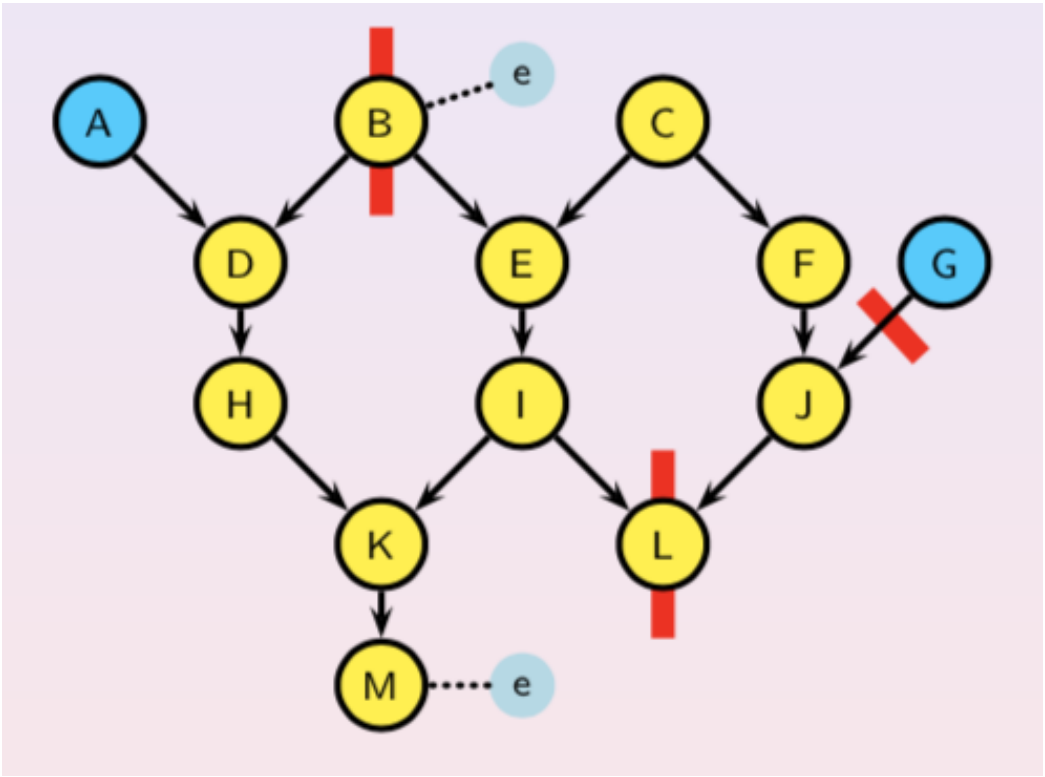
- Diverging connection



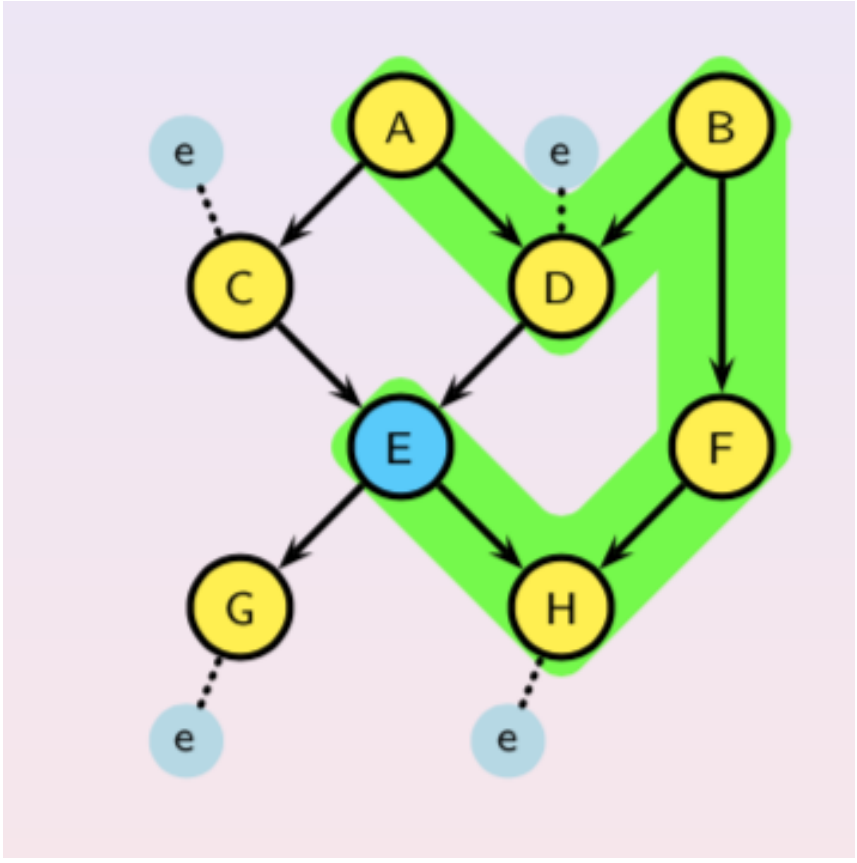
- Converging connection



d-separation example



d-separation example



d-separation example

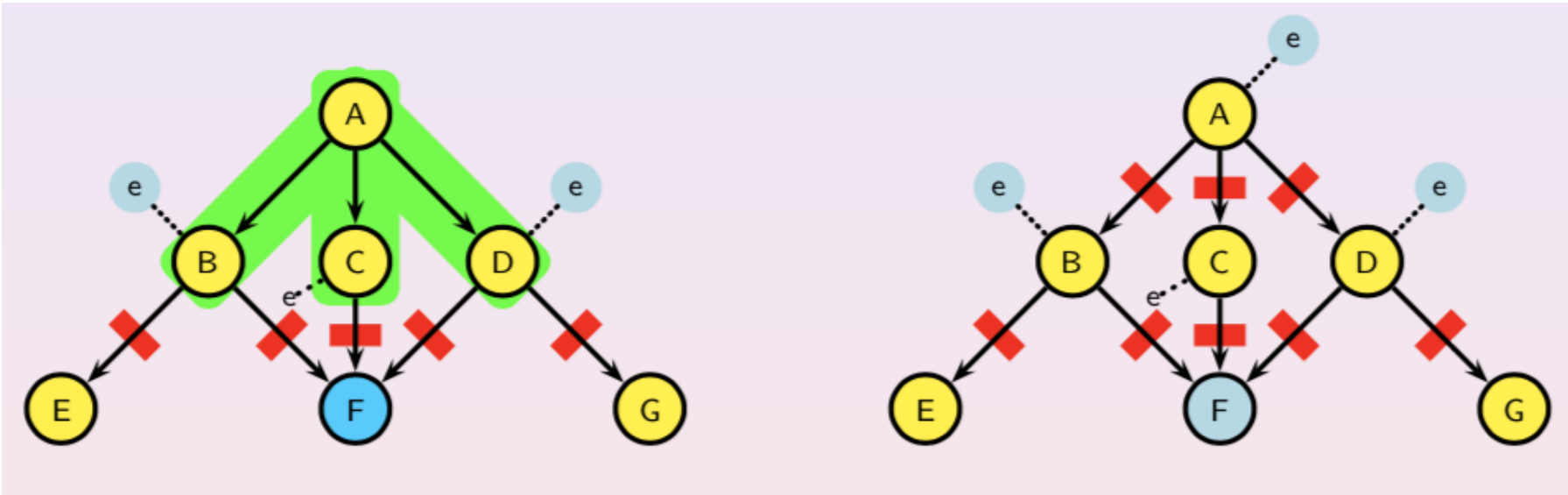
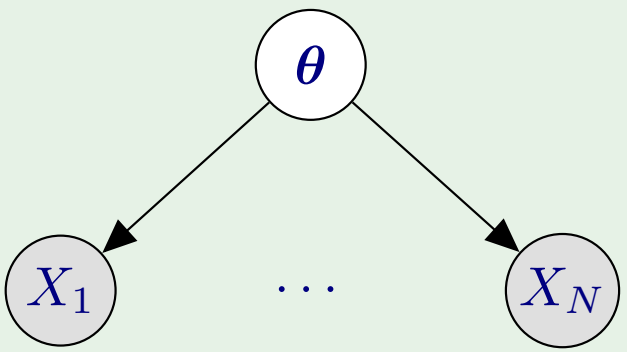


Plate notation

The idea is to avoid repeated substructures

Example: independent data points

- Assume the elements in a sample $X_1, \ldots, X_N$ are independent if the parameter θ is known



Unfolded notation

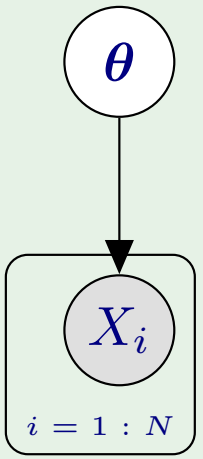


Plate notation

Plate notation

Example: linear regression

- $Y_i|\{\boldsymbol{w}, \boldsymbol{x}_i\} = \boldsymbol{w}^\intercal \boldsymbol{x}_i + \epsilon_i$ with $\boldsymbol{x}_i = [1, x_i]^\intercal$
- $\epsilon_i \sim \mathcal{N}(0, 1/\gamma)$ with γ known
- $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2})$

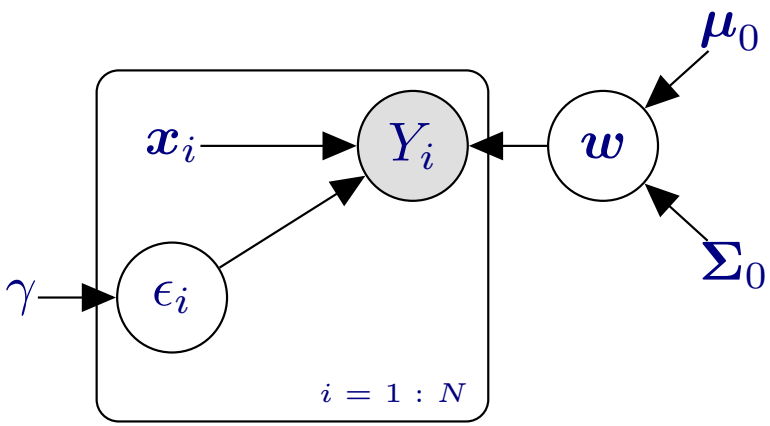
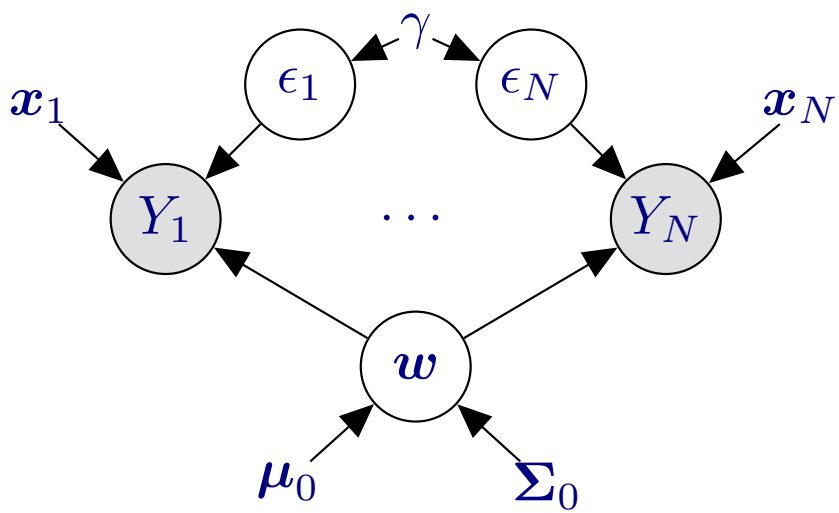


Plate notation



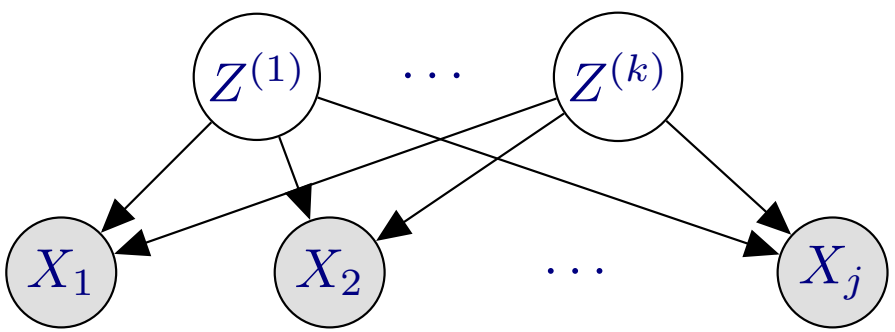
Unfolded notation

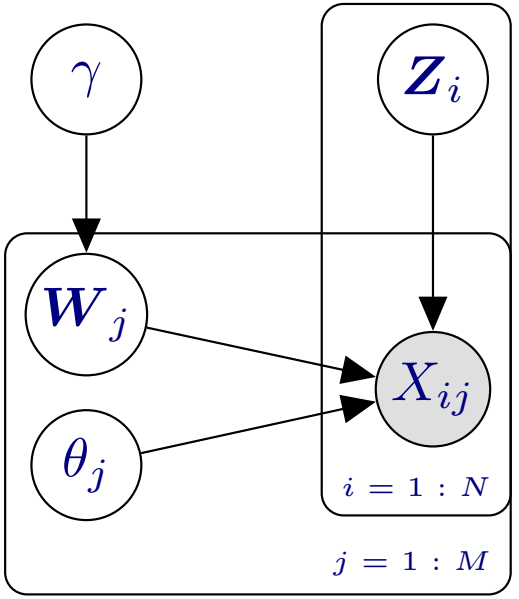
Latent variable models

Example: factor analysis model

- In general, latent variable models are regarded as probabilistic models where some of the variables cannot be observed
- FA summarizes a high-dimensional observation $\boldsymbol{X}$ of correlated variables by a smaller set of factors $\boldsymbol{Z}$ assumed independent a priori
- Model formulation:
  - $\boldsymbol{Z}_i \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I})$, $i = 1, \ldots, N$
  - $X_{i,j} | \{\boldsymbol{z}_i, \boldsymbol{w}_j, \theta_j\} \sim \mathcal{N}(\boldsymbol{w}_j^{\mathsf{T}} \boldsymbol{z}_i, 1/\theta_j)$, $i = 1, \ldots, N$, $j = 1, \ldots, M$

Local model

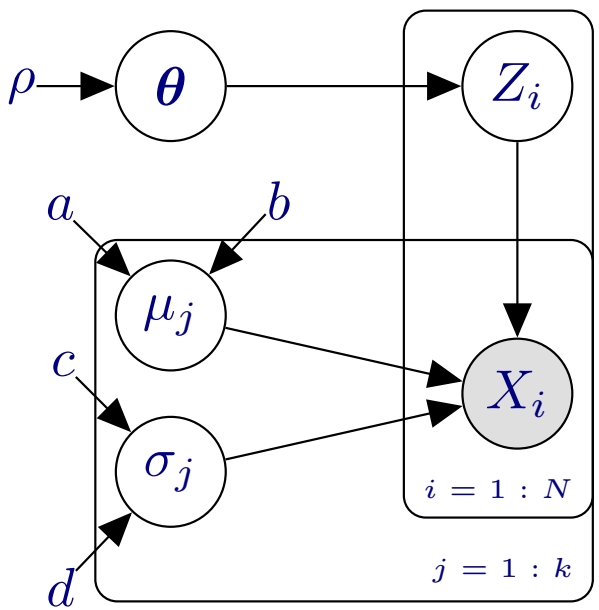




Latent variable models

Example: mixture of Gaussians

- Popular model for clustering
- Model formulation:
  - k Gaussians with frequencies $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$
  - N observations generated by
    - ⋆ $Z_i \sim \text{Multinom}(\boldsymbol{\theta})$, $i = 1, \ldots, N$
    - ⋆ $X_i | z_i \sim \mathcal{N}(\mu_i, \sigma_i)$, $i = 1, \ldots, N$
    - ⋆ Bayesian setting: priors on the parameters



Inference in Bayesian networks

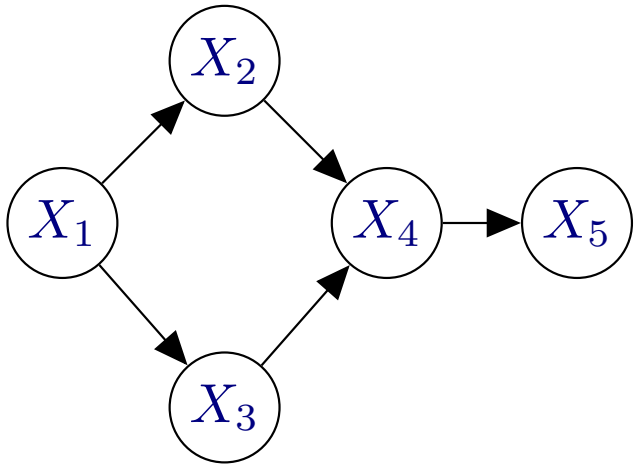
Assume a Bayesian network over variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$

$$\left.\begin{array}{r} \text{Bayesian network} \\ + \\ \text{Evidence } (\boldsymbol{X}_E) \end{array}\right\} \Rightarrow P(\boldsymbol{X}_I | \boldsymbol{X}_E)?$$

Inference methods

- Exact
  - Brute force: compute $P(\boldsymbol{X}, \boldsymbol{X}_E)$ and marginalize out $\boldsymbol{X} \setminus \boldsymbol{X}_I$
  - Take advantage of the network structure
- Approximate
  - Sampling
  - Deterministic

Exact inference: Variable elimination



- We are interested in X_5
- All variables are discrete
- $E = \emptyset$

$$
\begin{aligned}
p(x_5) &= \sum_{x_1,\ldots,x_4} p(x_1, x_2, x_3, x_4, x_5) \\
&= \sum_{x_1,\ldots,x_4} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_4) \\
&= \sum_{x_2,\ldots,x_4} \sum_{x_1} p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_4) \\
&= \sum_{x_2,\ldots,x_4} p(x_4|x_2, x_3)p(x_5|x_4) \sum_{x_1} p(x_1)p(x_2|x_1)p(x_3|x_1) \\
&= \sum_{x_2,\ldots,x_4} p(x_4|x_2, x_3)p(x_5|x_4)h(x_2, x_3)
\end{aligned}
$$

We have reached a similar problem as initially, but with one variable less.

Variable elimination algorithm

Input

$\mathcal{P}$: Conditional distributions in the network
$\boldsymbol{X}$: Variables in the network
X_I: Target variable
$\boldsymbol{X}_E$: Observed variables
$\boldsymbol{Y}$: All variables in $\boldsymbol{X}$ except X_I ($\boldsymbol{Y} = \boldsymbol{X} \setminus \{X_I\}$).

1. Restrict all the distributions in $\mathcal{P}$ to the evidence $\boldsymbol{X}_E = \boldsymbol{x}_E$
2. For each $Y \in \boldsymbol{Y}$,
   1. Let $\mathcal{P}_Y$ be the set of distributions in $\mathcal{P}$ that contain Y.
   2. $q := \prod_{p \in \mathcal{P}_Y} p$.
   3. $r := \sum_y q$.
   4. $\mathcal{P} := \mathcal{P} \setminus \mathcal{P}_Y \cup \{r\}$.
3. $p(x_I) = \prod_{p \in \mathcal{P}} p$.
4. Normalize p

Variable elimination algorithm

Input

$\mathcal{P}$: Conditional distributions in the network
$\boldsymbol{X}$: Variables in the network
X_I: Target variable
$\boldsymbol{X}_E$: Observed variables
$\boldsymbol{Y}$: All variables in $\boldsymbol{X}$ except X_I ($\boldsymbol{Y} = \boldsymbol{X} \setminus \{X_I\}$).

1. Restrict all the distributions in $\mathcal{P}$ to the evidence $\boldsymbol{X}_E = \boldsymbol{x}_E$
2. For each $Y \in \boldsymbol{Y}$
   1. Let $\mathcal{P}_Y$ be the set of distributions in $\mathcal{P}$ that contain Y
   2. $q := \prod_{p \in \mathcal{P}_Y} p \implies$ **COMPLEXITY**
   3. $r := \sum_y q$
   4. $\mathcal{P} := \mathcal{P} \setminus \mathcal{P}_Y \cup \{r\}$
3. $p(x_I) = \prod_{p \in \mathcal{P}} p$
4. Normalize p

Considerations about exact inference

- Product of functions raises complexity
  - Exponentially in the case of discrete variables
- Complexity also depends on the elimination order
- Representation of densities turns out to be relevant
  - Closed-form solutions to product and marginalization are preferable

Monte Carlo inference algorithms

- A Bayesian network is a representation of a joint probability distribution over $\boldsymbol{X} \Rightarrow$ it describes some population consisting of all the possible configurations of $\boldsymbol{X}$
- If the entire population was available, the inference problem could be solved exactly, basically by counting cases
- **Problem:** Population size can be huge or even infinite.
- Monte Carlo methods operate by drawing an artificial **sample** from it using some random mechanism
- The sample (much smaller than the population), is used to estimate the distribution of each variable of interest.

Key issues in a Monte Carlo inference algorithm:

1. The sampling mechanism
2. The functions (estimators) which compute the probabilities from the sample

Monte Carlo inference algorithms

- A Bayesian network is a representation of a joint probability distribution over $\boldsymbol{X} \Rightarrow$ it describes some population consisting of all the possible configurations of $\boldsymbol{X}$
- If the entire population was available, the inference problem could be solved exactly, basically by counting cases
- **Problem:** Population size can be huge or even infinite.
- Monte Carlo methods operate by drawing an artificial **sample** from it using some random mechanism
- The sample (much smaller than the population), is used to estimate the distribution of each variable of interest.

Key issues in a Monte Carlo inference algorithm:

1. The sampling mechanism
2. The functions (estimators) which compute the probabilities from the sample

Importance sampling. General setting

- Assume we have a random variable X with density $p(x)$
- Importance sampling is a technique designed for estimating the expected value of a function $f(X)$. It is based on the following transformation:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int \frac{p(x)}{p^*(x)} f(x)p^*(x)dx = \mathbb{E}_{p^*}\left[\frac{p(x)}{p^*(x)} f(x)\right],$$

  where p^* is a density function called the sampling or proposal distribution, s.t. $p^*(x) > 0$ whenever $p(x) > 0$.
- Therefore, $\mathbb{E}_p[f(x)]$ can be estimated by drawing a sample $x^{(1)}, \ldots, x^{(m)}$ from p^* and computing

$$\hat{\mathbb{E}}_p[f(x)] = \frac{1}{m} \sum_{j=1}^{m} \frac{p(x^{(j)})}{p^*(x^{(j)})} f(x^{(j)}),$$

  which is specially convenient if p^* is easier to handle than p

Importance sampling. General setting

- Assume we have a random variable X with density $p(x)$
- Importance sampling is a technique designed for estimating the expected value of a function $f(X)$. It is based on the following transformation:

$$\mathbb{E}_p[f(x)] = \int f(x)p(x)dx = \int \frac{p(x)}{p^*(x)} f(x)p^*(x)dx = \mathbb{E}_{p^*}\left[\frac{p(x)}{p^*(x)} f(x)\right],$$

  where p^* is a density function called the sampling or proposal distribution, s.t. $p^*(x) > 0$ whenever $p(x) > 0$.
- Therefore, $\mathbb{E}_p[f(x)]$ can be estimated by drawing a sample $x^{(1)}, \ldots, x^{(m)}$ from p^* and computing

$$\hat{\mathbb{E}}_p[f(x)] = \frac{1}{m} \sum_{j=1}^{m} \frac{p(x^{(j)})}{p^*(x^{(j)})} f(x^{(j)}),$$

  which is specially convenient if p^* is easier to handle than p

Importance sampling in probabilistic inference

- Consider a rather general setting, with continuous target variable X_i, observed variables $\boldsymbol{X}_E$, unobserved discrete variables $\boldsymbol{X}_D$ and unobserved continuous variables $\boldsymbol{X}_C$
- Assume we are interested in the probability of the variable taking values in a given interval (a, b). This amounts to computing

$$p(X_i \in (a,b)|\boldsymbol{x}_E) = \frac{\int_a^b \sum_{\boldsymbol{x}_D \in \Omega_{\boldsymbol{X}_D}} \int_{\boldsymbol{x}_C \in \Omega_{\boldsymbol{X}_C}} p(x_i, \boldsymbol{x}_C, \boldsymbol{x}_D, \boldsymbol{x}_E) \mathrm{d}\boldsymbol{x}_C \mathrm{d}x_i}{\sum_{\boldsymbol{x}_{D_i} \in \Omega_{\boldsymbol{X}_{D_i}}} \int_{\boldsymbol{x}_{C_i} \in \Omega_{\boldsymbol{X}_{C_i}}} p(\boldsymbol{x}_{C_i}, \boldsymbol{x}_{D_i}, \boldsymbol{x}_E) \mathrm{d}\boldsymbol{x}_{C_i}}$$

Importance sampling in probabilistic inference

Importance sampling can be applied to probabilistic inference in BNs taking into account that we can write $p(X_i \in (a,b)|\boldsymbol{x}_E)$ as

$$\begin{aligned} p(X_i \in (a,b)|\boldsymbol{x}_E) &= \mathbb{E}_{p(x_i|\boldsymbol{x}_E)}\left[\mathbf{1}_{(a,b)}(x_i)\right] \\ &= \mathbb{E}_{p^*}\left[\frac{p(x_i|\boldsymbol{x}_E)}{p^*(x_i)}\mathbf{1}_{(a,b)}(x_i)\right] \\ &= \frac{1}{p(\boldsymbol{x}_E)}\mathbb{E}_{p^*}\left[\frac{p(x_i,\boldsymbol{x}_E)}{p^*(x_i)}\mathbf{1}_{(a,b)}(x_i)\right], \end{aligned}$$

where $\mathbf{1}_{(a,b)}$ is the indicator function on interval (a,b).

Importance sampling in probabilistic inference

Denoting $g(x_i) = p(x_i, \boldsymbol{x}_E)$,

$$\begin{aligned}\hat{p}(X_i \in (a,b)|\boldsymbol{x}_E) &= \frac{1}{p(\boldsymbol{x}_E)} \hat{\mathbb{E}}_{p^*}\left[\frac{g(x_i)}{p^*(x_i)} \mathbf{1}_{(a,b)}(x_i)\right] \\ &= \frac{1}{p(\boldsymbol{x}_E)} \frac{1}{m} \sum_{j=1}^{m} \frac{g(x_i^{(j)})}{p^*(x_i^{(j)})} \mathbf{1}_{(a,b)}(x_i^{(j)})\end{aligned}$$

The normalizing constant, $p(\boldsymbol{x}_E)$, can be estimated using the same sample $x_i^{(1)}, \ldots, x_i^{(m)}$, just by summing all the *weights*:

$$\hat{p}(\boldsymbol{x}_E) = \sum_{j=1}^{m} \frac{g(x_i^{(j)})}{p^*(x_i^{(j)})}$$

Importance sampling in probabilistic inference

- The crucial step of Importance Sampling is the choice of the proposal distribution
- A careless choice can even lead to discarding most of the samples

Example

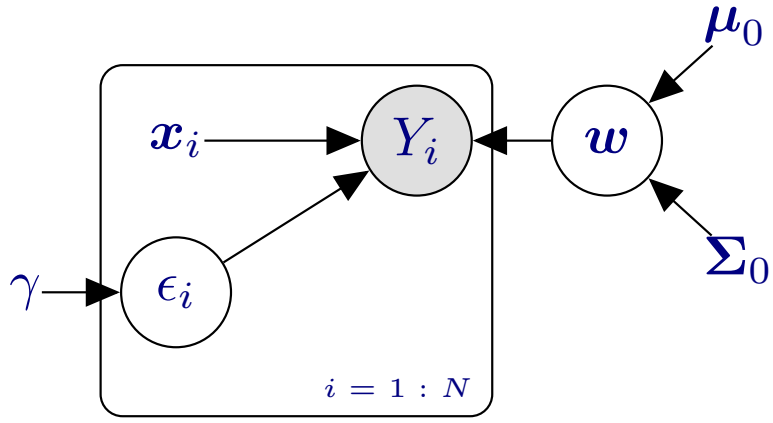
Consider binary variables X_1 and X_2 with $P(X_1 = 0) = 0.9$, $P(X_2 = 0|X_1 = 0) = 0$ and $P(X_2 = 0|X_1 = 1) = 0.5$. Assume X_2 is observed to be equal to 0. We have freedom to choose p^*.

- We choose p^* to be uniform $\implies > 50\%$ **of the samples are discarded (they have weight equal to 0)**
- We choose p^* to be $P(X_1) \implies > 90\%$ **of the samples are discarded**

Importance sampling. Example

Linear regression

- $Y_i|\{\boldsymbol{w}, \boldsymbol{x}_i\} = \boldsymbol{w}^\intercal \boldsymbol{x}_i + \epsilon_i$ with $\boldsymbol{x}_i = [1, x_i]^\intercal$
- $\epsilon_i \sim \mathcal{N}(0, 1/\gamma)$ with γ known
- $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2})$



- We have to compute

$$p(\boldsymbol{w}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

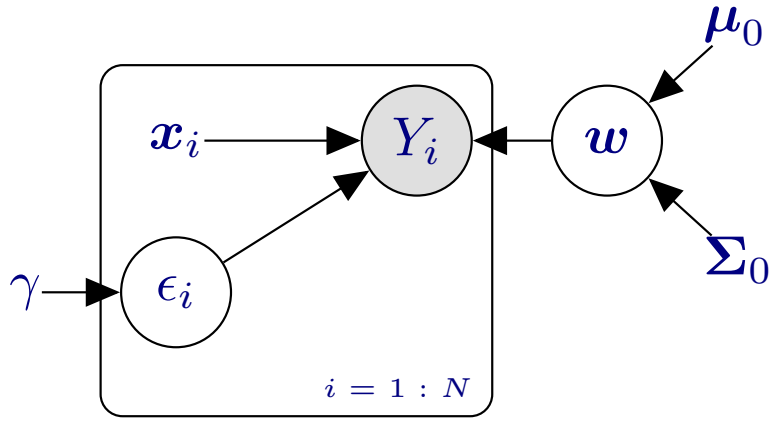
- $\boldsymbol{\mu} = (\mathbb{E}[w_0], \mathbb{E}[w_1])$
- $\Sigma = \begin{pmatrix} \sigma^2_{w_0} & \sigma_{w_0,w_1} \\ \sigma_{w_0,w_1} & \sigma^2_{w_1} \end{pmatrix}$

- We have to sample ϵ_i and $\boldsymbol{w}$ from p^*
- We can assume, for instance p^* to be the product of three independent standard normals for ϵ_i, w_0 and w_1

Importance sampling. Example

Linear regression

- $Y_i|\{\boldsymbol{w}, \boldsymbol{x}_i\} = \boldsymbol{w}^\intercal \boldsymbol{x}_i + \epsilon_i$ with $\boldsymbol{x}_i = [1, x_i]^\intercal$
- $\epsilon_i \sim \mathcal{N}(0, 1/\gamma)$ with γ known
- $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_0 = \mathbf{0}, \boldsymbol{\Sigma}_0 = \mathbf{I}_{2\times 2})$



- We have to compute

$$p(\boldsymbol{w}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

- $\boldsymbol{\mu} = (\mathbb{E}[w_0], \mathbb{E}[w_1])$
- $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_{w_0} & \sigma_{w_0,w_1} \\ \sigma_{w_0,w_1} & \sigma^2_{w_1} \end{pmatrix}$

- We have to sample ϵ_i and $\boldsymbol{w}$ from p^*
- We can assume, for instance p^* to be the product of three independent standard normals for ϵ_i, w_0 and w_1

Importance sampling. Example

Taking into account that

- $\sigma^2_{w_k} = \mathbb{E}[w_k^2] - (\mathbb{E}[w_k])^2$, $k = 0, 1$
- $\sigma_{w_0, w_1} = \mathbb{E}[w_0 w_1] - \mathbb{E}[w_0]\mathbb{E}[w_1]$

it turns out that what we have to estimate to solve the linear regression problem is

- $\mathbb{E}[w_0], \mathbb{E}[w_1]$, $\mathbb{E}[w_0^2], \mathbb{E}[w_1^2]$ and $\mathbb{E}[w_0 w_1]$
- Note: all expectations are taken w.r.t. $p(\boldsymbol{w}|\boldsymbol{x})$

Importance sampling. Example

1. Generate a sample $\boldsymbol{z}_j = (\epsilon_i^{(j)}, w_0^{(j)}, w_1^{(j)}, y_i^{(j)})$, $j = 1, \ldots, M$ from p^*
2. Compute the estimations:

$$\hat{\mathbb{E}}[w_0] = \frac{1}{\hat{p}(\boldsymbol{x})} \frac{1}{M} \sum_{j=1}^{M} \frac{g(\boldsymbol{z}_j)}{p^*(\boldsymbol{z}_j)} w_0^{(j)}$$

$$\hat{\mathbb{E}}[w_1] = \frac{1}{\hat{p}(\boldsymbol{x})} \frac{1}{M} \sum_{j=1}^{M} \frac{g(\boldsymbol{z}_j)}{p^*(\boldsymbol{z}_j)} w_1^{(j)}$$

$$\hat{\mathbb{E}}[w_0^2] = \frac{1}{\hat{p}(\boldsymbol{x})} \frac{1}{M} \sum_{j=1}^{M} \frac{g(\boldsymbol{z}_j)}{p^*(\boldsymbol{z}_j)} {w_0^{(j)}}^2$$

$$\hat{\mathbb{E}}[w_1^2] = \frac{1}{\hat{p}(\boldsymbol{x})} \frac{1}{M} \sum_{j=1}^{M} \frac{g(\boldsymbol{z}_j)}{p^*(\boldsymbol{z}_j)} {w_1^{(j)}}^2$$

$$\hat{\mathbb{E}}[w_0 w_1] = \frac{1}{\hat{p}(\boldsymbol{x})} \frac{1}{M} \sum_{j=1}^{M} \frac{g(\boldsymbol{z}_j)}{p^*(\boldsymbol{z}_j)} w_0^{(j)} w_1^{(j)}$$

Markov Chain Monte Carlo (MCMC)

- We start with an initial configuration of the variables in the network
- A new item in the sample is generated conditional on the previous configuration
- The elements of the sample are no longer independent, instead they form a **Markov chain**

Gibbs sampling

A new item in the sample is generated by simulating each variable at a time conditional on the value of the other variables sampled so far:

- Assume $\boldsymbol{X} = \{X_1, \ldots, X_n\}$
- Let $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$ be the current configuration
- A new configuration, $\boldsymbol{x}^{(i+1)}$ is generated by sampling each variable X_k, $k = 1, \ldots, n$ with

$$p(x_j^{(i+1)} | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$$

Markov Chain Monte Carlo (MCMC)

- We start with an initial configuration of the variables in the network
- A new item in the sample is generated conditional on the previous configuration
- The elements of the sample are no longer independent, instead they form a **Markov chain**

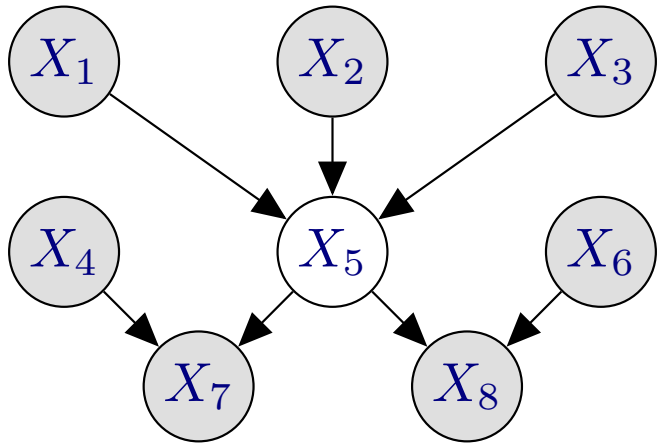
Gibbs sampling

A new item in the sample is generated by simulating each variable at a time conditional on the value of the other variables sampled so far:

- Assume $\boldsymbol{X} = \{X_1, \ldots, X_n\}$
- Let $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$ be the current configuration
- A new configuration, $\boldsymbol{x}^{(i+1)}$ is generated by sampling each variable X_k, $k = 1, \ldots, n$ with

$$p(x_j^{(i+1)} | x_1^{(i+1)}, \ldots, x_{j-1}^{(i+1)}, x_{j+1}^{(i)}, \ldots, x_n^{(i)})$$

Markov Chain Monte Carlo (MCMC)



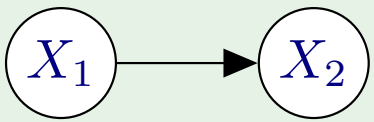
$$p^*(X_5) \propto p(X_5|X_1, X_2, X_3)p(X_7|X_4, X_5)p(X_8|X_5, X_6)$$

Problems

- Generating an initial configuration with positive probability can be hard
- The dependence between elements of the sample can make it converge slowly

Markov Chain Monte Carlo (MCMC)

Example



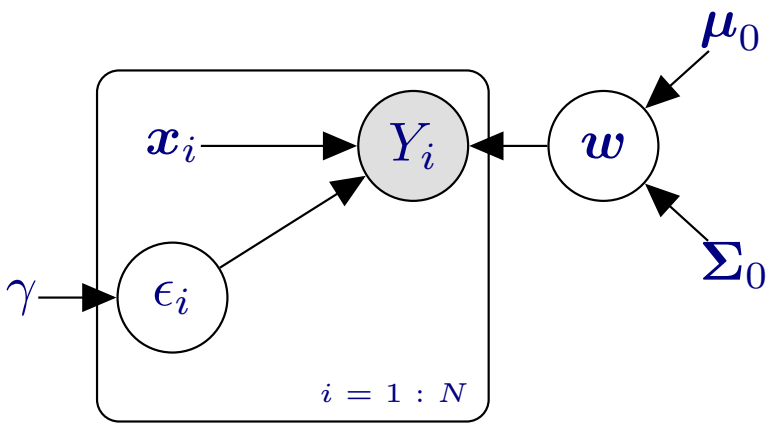
$$p(x_2|x_1) = p(\bar{x}_2|\bar{x}_1) = \delta \approx 1, \quad p(x_1) = 0.5$$

- Assume $X_1 = x_1 \Rightarrow p(x_2|w_{x_2}) = p(x_2|x_1) = \delta$
- If we draw $X_2 = x_2$, then X_1 is sampled from

$$\begin{aligned} p(x_1|w_{x_1}) &= p(x_1|x_2) = \frac{p(x_2|x_1)p(x_1)}{p(x_2|x_1)p(x_1) + p(x_2|\bar{x}_1)p(\bar{x}_1)} \\ &= \frac{\delta \times 0.5}{\delta \times 0.5 + (1-\delta) \times 0.5} = \delta \end{aligned}$$

- Hence, we obtain $(X_1 = x_1, X_2 = x_2)$ with probability ≈ 1, and, if a variable ever changed its value, the other one would change as well, thus appearing many times the configuration $(X_1 = \bar{x}_1, X_2 = \bar{x}_2)$.

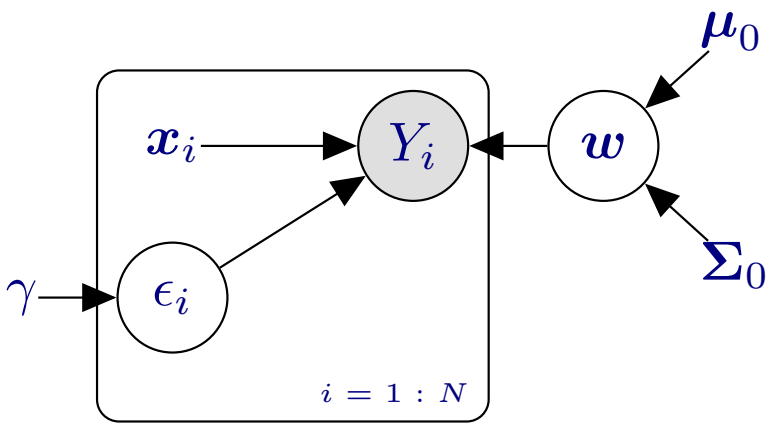
Gibbs sampling example: Linear regression



1. Start with an initial random configuration
2. Simulate a sample by changing coordinates one at a time
3. Estimate $\mathbb{E}[w_0], \mathbb{E}[w_1]$, $\mathbb{E}[w_0^2], \mathbb{E}[w_1^2]$ and $\mathbb{E}[w_0 w_1]$ using sample mean estimators

You still need to be able to compute the distribution of each variable given its Markov blanket

Gibbs sampling example: Linear regression



1. Start with an initial random configuration
2. Simulate a sample by changing coordinates one at a time
3. Estimate $\mathbb{E}[w_0], \mathbb{E}[w_1]$, $\mathbb{E}[w_0^2], \mathbb{E}[w_1^2]$ and $\mathbb{E}[w_0 w_1]$ using sample mean estimators

You still need to be able to compute the distribution of each variable given its Markov blanket

Inference as an optimization problem

General setting

- We want to approximate $p(\boldsymbol{z}|\boldsymbol{x})$
- The idea here is to find the distribution $q(\boldsymbol{z}|\boldsymbol{x})$ ($= q(\boldsymbol{z})$ for short) that best approximates $p(\boldsymbol{z}|\boldsymbol{x})$ within a set of ditributions $\mathcal{Q}$, according to some optmality criterium
- In other words, if we have a functional δ that measures how different two distributions are, what we are looking for is

$$\hat{q}(\boldsymbol{z}) = \arg\min_{q \in \mathcal{Q}} \delta(q(\boldsymbol{z}), p(\boldsymbol{z}|\boldsymbol{x}))$$

Inference as an optimization problem

KL divergence

- A popular choice is Kullback-Leibler divergence

$$\mathrm{KL}\,(\,f\,\|\,g\,) = \int_{\boldsymbol{z}} f(\boldsymbol{z}) \log \frac{f(\boldsymbol{z})}{g(\boldsymbol{z})} = \mathbb{E}_f \left[\log \frac{f(\boldsymbol{z})}{g(\boldsymbol{z})} \right]$$

- Note that it is not symmetric

Alternative KL projections

Information-projection

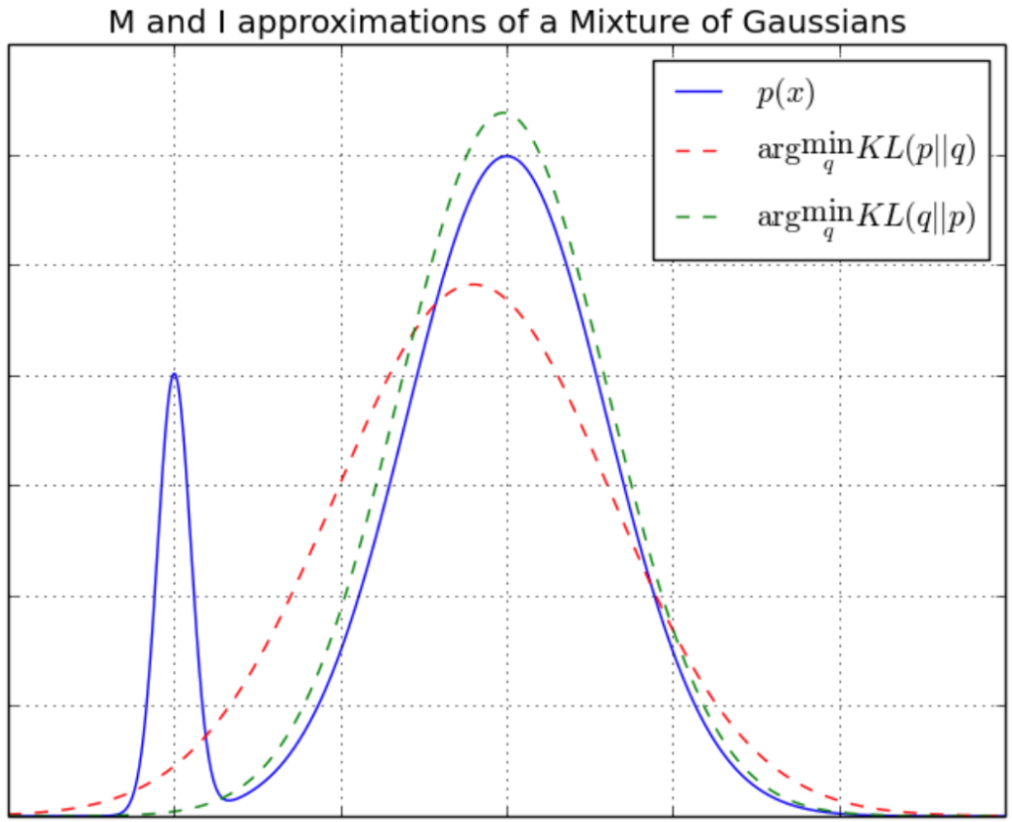
- Minimizes $\mathrm{KL}\,(q||p)$, that is, the expected information loss under q if p is used instead of q.
- Factorizes as $\mathrm{KL}\,(q||p) = -\,\mathbb{E}_q[\ln p(\mathbf{z})] - \mathcal{H}_q$.
- Preference given to q that has:
  - High probability at p-probable regions.
  - Very small probability given to any region where p is small.
  - High entropy ("large variance")

Moment-projection

- Minimizes $\mathrm{KL}\,(p||q)$, that is, the expected information loss under p if q is used instead of p.
- Factorizes as $\mathrm{KL}\,(p||q) = -\,\mathbb{E}_p[\ln q(\mathbf{z})] - \mathcal{H}_p$.
- Preference given to q that has:
  - High probability allocated to regions that are probable according to p.
  - Non-zero probability given to any region with p is non-negligible.
  - No explicit focus of entropy

$$\mathcal{H}_f = -\int_{\boldsymbol{z}} f(\boldsymbol{z}) \log f(\boldsymbol{z}) d\boldsymbol{z}$$

Alternative KL projections



- Moment projection – optimizing $\mathrm{KL}\,(p||q)$ – has slightly larger variance.
- Similar means, but Information projection – optimizing $\mathrm{KL}\,(q||p)$ – focuses mainly on the most prominent mode.

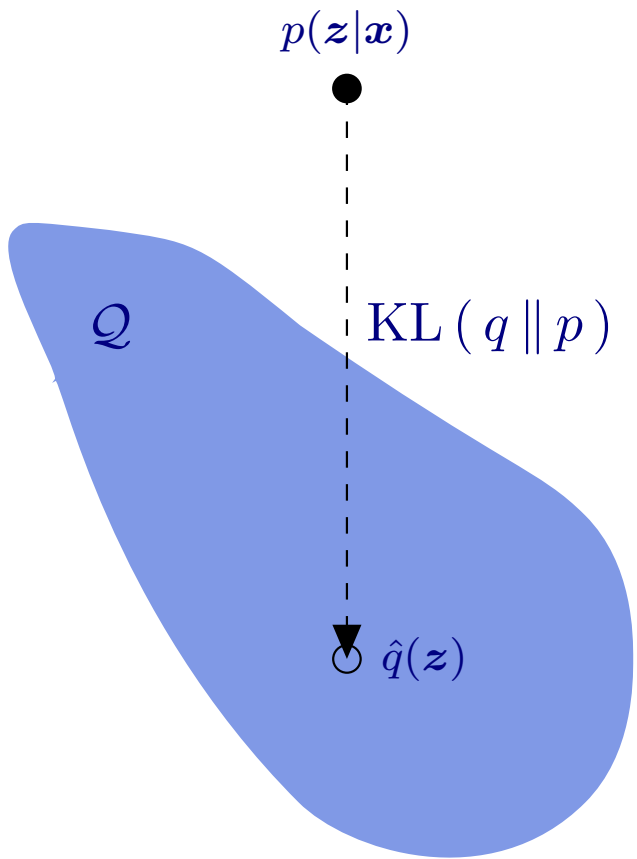
Variational inference setup

Approximation

Variational inference approximates $p(\boldsymbol{z}|\boldsymbol{x})$ by

$$\hat{q}(\boldsymbol{z}) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}|\boldsymbol{x}) \right)$$

- With a target-set $\mathcal{Q}$ and a divergence function $\mathrm{KL}\left(q \,\|\, p \right)$ we can think of the approximation as a projection of p onto the sub-space defined by $\mathcal{Q}$.
- The projection ends up at $\hat{q}$, which minimizes $\mathrm{KL}\left(q \,\|\, p \right)$ over all $q \in \mathcal{Q}$.
- We should choose $\mathcal{Q}$ so that it is flexible enough to capture "most of" $p(\cdot)$, meaning that $\mathrm{KL}\left(q \,\|\, p \right)$ is "typically small".



ELBO: Evidence Lower-BOund

Notice how we can rearrange the KL divergence as follows:

$$\mathrm{KL}\left(\, q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}|\boldsymbol{x}) \,\right) = \mathbb{E}_q\left[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}\right] = \mathbb{E}_q\left[\log \frac{q(\boldsymbol{z}) \cdot p(\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x}) \cdot p(\boldsymbol{x})}\right]$$

$$= \log p(\boldsymbol{x}) - \left(-\mathbb{E}_q\left[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z},\boldsymbol{x})}\right]\right) = \log p(\boldsymbol{x}) - \mathcal{L}\left(q\right)$$

where the Evidence Lower Bound (ELBO) is $\mathcal{L}\left(q\right) = -\mathbb{E}_q\left[\log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z},\boldsymbol{x})}\right]$.

Variational inference focuses on ELBO:

$$\log p(\boldsymbol{x}) = \mathcal{L}\left(q\right) + \mathrm{KL}\left(\, q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}|\boldsymbol{x}) \,\right)$$

Since $\log p(\boldsymbol{x})$ is constant wrt. q and $\mathrm{KL}\left(\, q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}|\boldsymbol{x}) \,\right) \geq 0$ it follows:

- We can minimize $\mathrm{KL}\left(\, q(\boldsymbol{z}) \,\|\, p(\boldsymbol{z}|\boldsymbol{x}) \,\right)$ by maximizing $\mathcal{L}\left(q\right)$
- This is **computationally simpler** because it uses $p(\boldsymbol{z}, \boldsymbol{x})$ instead of $p(\boldsymbol{z}|\boldsymbol{x})$.
- $\mathcal{L}\left(q\right)$ is a *lower bound* of the marginal data log likelihood $\log p(\boldsymbol{x})$.

$\rightsquigarrow$ During inference, we will look for $\hat{q}(\boldsymbol{z}) = \arg\max_{q \in \mathcal{Q}} \mathcal{L}\left(q\right)$.

Conclusions

- PGMs provide a well founded way of handling uncertainty
- From a Bayesian point of view, inference and learning are connected tasks
- If scalability is on the focus, VI and IS are preferable to MCMC
- Interpretability is a key issue