Variational Inference

Introduction: Bayesian networks

Helge Langseth and Thomas Dyhre Nielsen*

Oct. 2018

* Some of the introductory slides are stolen from Manfred Jaeger, Aalborg University.

People involved

Universidad de Almería:

- Antonio Salmerón
- Andrés Masegosa
- Rafael Cabañas de Paz
- Rafael Rumí
- Darío Ramós-Lopez

Aalborg University:

- Thomas D. Nielsen
- Anders L. Madsen

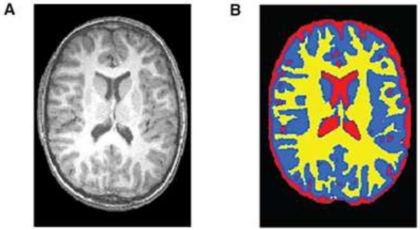
NTNU:

- Helge Langseth

Example 1: SLAM

Simultaneous Localization and Mapping: learn a map of the environment and locate current position

Example 2: Image Segmentation



(source: http://pubs.niaaa.nih.gov/publications/arh313/243-246.htm)

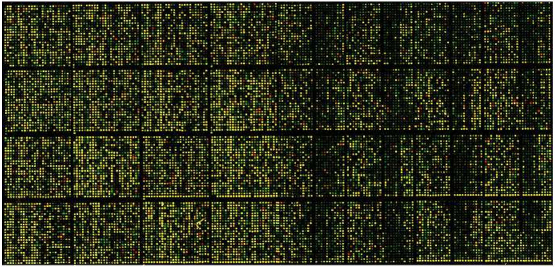
- Divide image into small number of regions representing structurally similar areas

Example 3: Statistical Semantics

Given a collection of texts:

Goal: automatically learn semantic descriptors for documents and words, that support document clustering, text understanding, information retrieval ...

Micro-array gene expression data:



Which genes are expressed under which conditions? Which are co-regulated, or functionally dependent?

Common ground in 4 examples:

- Use a **probabilistic model**, typically learned from data (using **statistical learning techniques**)
- Apply probabilistic inference algorithms to use models for prediction (**classification**, **regression**), structure analysis (**clustering**, **segmentation**)

Advantages of probabilistic/statistical methods:

- Principled quantification of prediction uncertainties
- Robust and principled techniques to deal with incomplete information, missing data.

Need: probabilistic models that . . .

- can represent models for high-dimensional state spaces
- support efficient learning and inference techniques

Probabilistic Graphical Models . . .

- support a structured specification of high-dimensional distributions in terms of low-dimensional factors
- structured representation can be exploited for efficient learning and inference algorithms (sometimes ...)
- graphical representation gives human-friendly design and description possibilities

This seminar series is about efficient inference and learning for Bayesian networks using “Variational inference”.

Plan for this weeks

Day 1: Bayesian networks – Definition and inference

- Definition of Bayesian networks: Syntax and semantics
- Exact inference
- Approximate inference using MCMC

Day 2: Variational inference – Introduction and basis

- Approximate inference through the *Kullback-Leibler divergence*
- *Variational Bayes*
- The *mean-field* approach to Variational Bayes

Day 3: Variational Bayes – cont'd

- Solving the VB equations
- Introducing Exponential families

Day 4: Scalable Variational Bayes

- Variational message passing
- Stochastic gradient ascent
- Stochastic variational inference

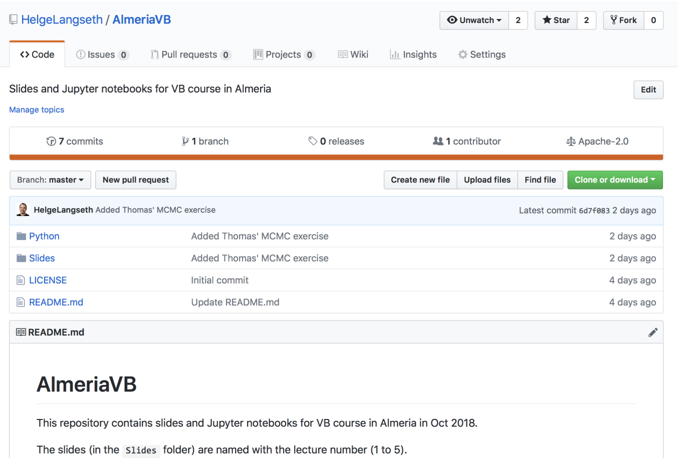
Day 5: Current approaches and extensions

- Variational Auto Encoders
- Black Box variational inference
- Probabilistic Programming Languages

Online repository

Materials for the course, including (almost last-version of) the slides, and description of implementation-tasks are available at

`https://github.com/HelgeLangseth/AlmeriaVB`



Starting-point

I will assume that these topics are (well) known:

- Probabilities, $P(X = x)$; conditional probabilities, $P(X = x \mid Y = y)$.
- Independence, $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}$; conditional independence $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$.
- "Standard" probability calculus:

Product: $P(x, y) = P(x \mid y) \cdot P(y) = P(x) \cdot P(y \mid x)$.

Sum-rule: $P(x \vee y) = P(x) + P(y) = P(x) - P(y \wedge x)$.

Total probability: $P(Y = y) = \sum_{x'} P(y \mid X = x') \cdot P(X = x')$.

Bayes rule: $P(x \mid y) = P(x) \cdot P(y \mid x) / P(y)$.

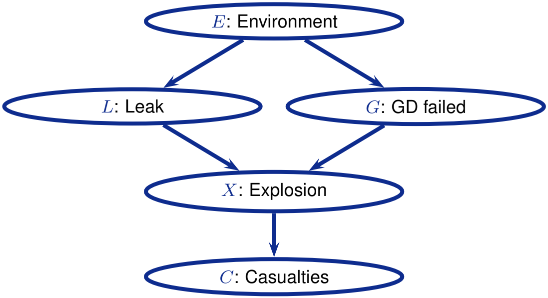
I will assume that these topics are (somewhat) known:

- Bayesian network syntax and semantics.
- Exact inference in Bayesian networks.
- Approximate inference using MCMC.

Implementation tasks:

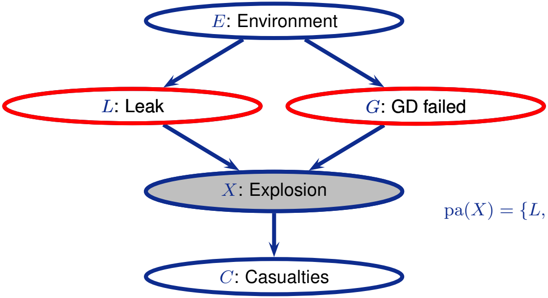
- We will get some implementation tasks as we move along.
- You will be supplied partly running Python-code (in the form of Jupyter notebooks).
- You will need to have **Python 3.x** on your computer (`https://www.python.org/downloads/`), and a set of packages (`numpy`, `scipy`, `matplotlib`, `jupyter`).

A simple example: “Explosion”



$P(E, L, G, X, C)$

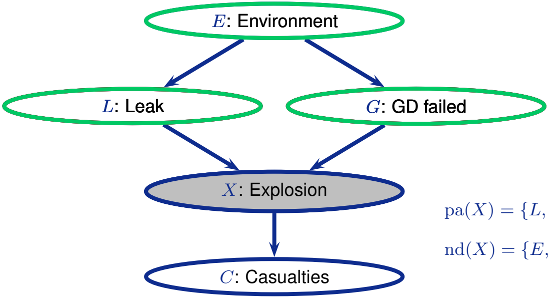
A simple example: “Explosion”



$$\mathrm{pa}(X) = \{L, G\}$$

$$P(E, L, G, X, C)$$

A simple example: “Explosion”

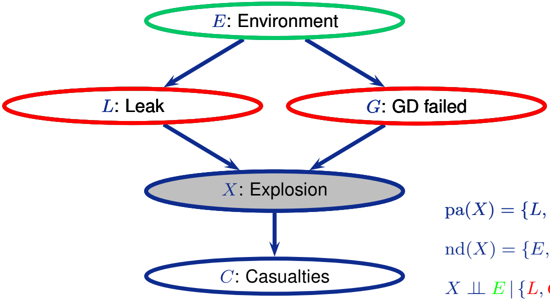


$\mathrm{pa}(X) = \{L, G\}$

$\mathrm{nd}(X) = \{E, L, G\}$

$P(E, L, G, X, C)$

A simple example: “Explosion”



$\mathrm{pa}(X) = \{L, G\}$

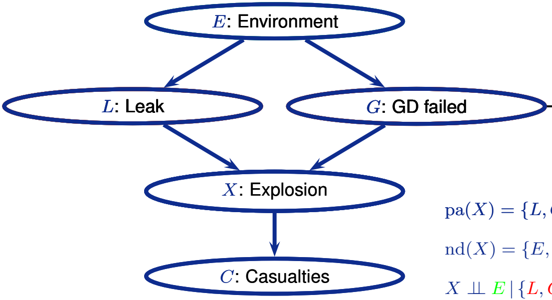
$\mathrm{nd}(X) = \{E, L, G\}$

$X \perp\!\!\!\perp E \,|\, \{L, G\}$

Other d-sep. rules: Jensen&Nielsen (07)

$P(E, L, G, X, C)$

A simple example: "Explosion"



G	$E=$hostile	$E=$normal
yes	$\lambda_H \cdot \tau/2$	$\lambda_N \cdot \tau/2$
no	$1-\lambda_H \cdot \tau/2$	$1-\lambda_N \cdot \tau/2$

$P(G \mid \mathrm{pa}(G))$

$\mathrm{pa}(X) = \{L, G\}$

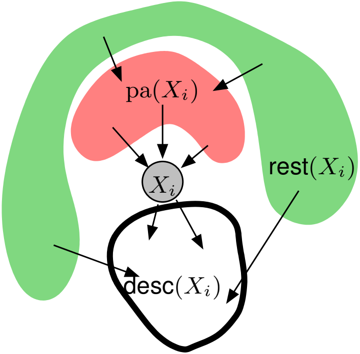
$\mathrm{nd}(X) = \{E, L, G\}$

$X \perp\!\!\!\perp E \mid \{L, G\}$ (Hence, $P(X \mid E, L, G) = P(X \mid L, G)$)

Other d-sep. rules: Jensen&Nielsen (07)

$$P(E,L,G,X,C) = P(E)\cdot P(L \mid E)\cdot P(G \mid E,L)\cdot P(X \mid E,L,G)\cdot P(C \mid E,L,G,X)$$
$$= P(E)\cdot P(L \mid E)\cdot P(G \mid E) \quad \cdot P(X \mid L,G) \quad \cdot P(C \mid X)$$

Markov properties $\Leftrightarrow$ Factorisation property



In the distribution P defined by the BN the following independence relation holds:

$$P(X_i \mid \mathrm{pa}(X_i), \text{rest}(X_i)) = P(X_i \mid \mathrm{pa}(X_i))$$

"rest" contains all nodes except:

- X_i itself
- X_i's parents
- X_i's descendants

"X_i is independent of its non-descendants given its parents"

Bayesian network syntax

Let $X_1, \ldots, X_n$ be a collection of random variables. A Bayesian network over $X_1, \ldots, X_n$ consists of

- a directed acyclic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ whose nodes $\mathcal{V}$ are the variables $X_1, \ldots, X_n$
- a set of local conditional distributions, $\mathcal{P} = \{p(X_i \mid \mathrm{pa}(X_i)), X_i \in \mathcal{V}\}$, where $\mathrm{pa}(X_i)$ are the parents of X_i in $\mathcal{G}$ as defined by the edges $\mathcal{E}$.

Bayesian network semantics

A Bayesian network $\mathcal{N}$ with nodes $X_1, \ldots, X_n$ defines a joint distribution

$$p_{\mathcal{N}}(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i \mid \mathrm{pa}(X_i))$$

Exact inference

Bayesian network inference

Inference in the Bayesian network amounts to calculating $p(\mathbf{Z} = \mathbf{z} \mid \mathbf{X} = \mathbf{x})$, where

- $\mathbf{X} \subset \mathcal{V}$ are the observed variables, currently taking the configuration $\mathbf{X} = \mathbf{x}$.
- $\mathbf{Z} \subseteq \mathcal{V} \setminus \mathbf{X}$ are our variables of interest.

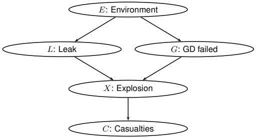
Inference is therefore the tool to answer any probabilistic query we may have in our domain, given a partial (or empty) observation from the domain

- *“What is the chance that Almería will move back to La Liga before 2020, given that they didn’t win during the first four games of this campaign?”*
- *“What is the probability of Google going bankrupt before you are done with your education?”*

Probabilistic inference - Example

Probability propagation

$$\left.\begin{array}{c}\text{Bayesian network}\\ +\\ \text{Evidence: } \mathbf{X}_E = \mathbf{x}_e\end{array}\right\} \Rightarrow P(\mathbf{X}_Q|\mathbf{x}_e)?$$

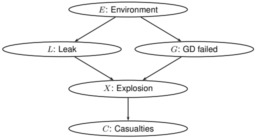


Example: Calculate $P(X = x \,|\, G = g)$

$$\begin{aligned}
P(x,g) &= \sum_e \sum_l \sum_c P(E=e, L=l, g, X=x, C=c) \\
&= \sum_e \sum_l \sum_c P(e)\cdot P(l\,|\,e)\cdot P(g\,|\,e)\cdot P(x\,|\,l,g)\cdot P(c\,|\,x) \\
&= \sum_e P(e)\cdot P(g\,|\,e) \sum_l P(l\,|\,e)\cdot P(x\,|\,l,g) \sum_c P(c\,|\,x)
\end{aligned}$$

Probability propagation

$$\left.\begin{array}{c}\text{Bayesian network}\\ +\\ \text{Evidence: } \mathbf{X}_E = \mathbf{x}_e\end{array}\right\} \Rightarrow P(\mathbf{X}_Q|\mathbf{x}_e)?$$



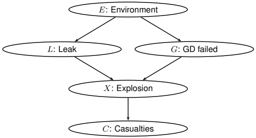
Example: Calculate $P(X = x \mid G = g)$

$$\begin{aligned} P(x, g) &= \sum_e P(e) \cdot P(g \mid e) \sum_l P(l \mid e) \cdot P(x \mid l, g) \\ P(X = x \mid G = g) &= \frac{P(x, g)}{\sum_{x'} P(X = x', g)} \propto P(x, g) \end{aligned}$$

Probabilistic inference - Example

Probability propagation

$$\left.\begin{array}{c}\text{Bayesian network}\\ +\\ \text{Evidence: } \mathbf{X}_E = \mathbf{x}_e\end{array}\right\} \Rightarrow P(\mathbf{X}_Q|\mathbf{x}_e)?$$



Operations to calculate $P(\mathbf{X}_Q \,|\, \mathbf{x}_e)$

Restriction: Restrict domain of a potential (e.g., $P(g|E)$ from $P(G|E)$)

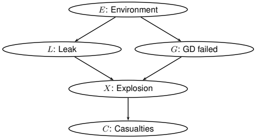
Combination: Multiplication of potentials (e.g., $P(l \,|\, e) \cdot P(X \,|\, l, g)$)

Marginalisation: Sum/integrate out a variable form a potential, e.g., the operation $\sum_l P(l \,|\, e) \cdot P(x \,|\, l, g)$, which removes L from the potential over $\{L, X, E, G\}$ and results in $P(x \,|\, e, g)$ over $\{X, E, G\}$.

Probabilistic inference - Example

Probability propagation

$$\left.\begin{array}{c}\text{Bayesian network}\\ +\\ \text{Evidence: } \mathbf{X}_E = \mathbf{x}_e\end{array}\right\} \Rightarrow P(\mathbf{X}_Q|\mathbf{x}_e)?$$



Requirements for efficient calculation of $P(\mathbf{X}_Q \mid \mathbf{x}_e)$

- Constraints wrt. structure:
  - Size of combined potentials
- Constraints wrt. distributions:
  - Ability to **perform** operations
  - Ability to **represent results** of operations

Computation of *conditional distributions*

Given $\mathbf{X} = X_1, \ldots, X_k \subset \mathcal{V}$, $x_i \in dom(X_i)$, $\mathbf{Z} = Z_1, \ldots, Z_l \subset \mathcal{V}$, compute

$$p(\mathbf{Z} \mid \mathbf{X} = \mathbf{x})$$

Especially: *Single variable posterior distributions*: for each $Z \notin \mathbf{X}$ compute $p(Z \mid \mathbf{X} = \mathbf{x})$.

Problem reduction

This problem can be reduced to the computation of partial distributions, i.e., functions

$$p(\mathbf{Z}, \mathbf{X} = \mathbf{x})$$

(function defined on $dom(\mathbf{Z})$) because

$$p(\mathbf{Z} \mid \mathbf{X} = \mathbf{x}) = p(\mathbf{Z}, \mathbf{X} = \mathbf{x}) / p(\mathbf{X} = \mathbf{x})$$

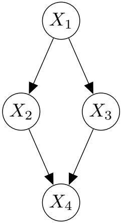
Direct approach: Denote $\mathbf{V}(= V_1, \ldots, V_m) := \mathcal{V} \setminus (\mathbf{X} \cup \mathbf{Z})$, and let $\mathbf{v}$ range over $dom(\mathbf{V})$. Then for $\mathbf{z} \in dom(\mathbf{Z})$:

$$
\begin{aligned}
p(\mathbf{Z} = \mathbf{z}, \mathbf{X} = \mathbf{x}) &= \sum_{\mathbf{v}} p(\mathbf{Z} = \mathbf{z}, \mathbf{X} = \mathbf{x}, \mathbf{V} = \mathbf{v}) \\
&= \sum_{\mathbf{v}} \prod_i p(Y_i \mid \mathrm{pa}(Y_i))(\mathbf{z}, \mathbf{x}, \mathbf{v}) \\
&= \sum_{v_1 \in dom(V_1)} \cdots \sum_{v_m \in dom(V_m)} \prod_i p(Y_i \mid \mathrm{pa}(Y_i))(\mathbf{z}, \mathbf{x}, \mathbf{v})
\end{aligned}
$$

"Algorithm": Sum out the v_j one by one, move factors $p(Y_i \mid \mathrm{pa}(Y_i))(\mathbf{z}, \mathbf{x}, \mathbf{v})$ that do not depend on current v_j (because $V_j \notin \{Y_i\} \cup \mathrm{pa}(Y_i)$) out of the sum.

Advantage: Can be used to compute conditional distributions for arbitrary set $\mathbf{Z}$ of query variables.

Example



	X_1 = t	X_1 = f
	.5	.5

X_1	X_2 = t	X_2 = f
t	.7	.3
f	.1	.9

X_1	X_3 = t	X_3 = f
t	.7	.3
f	.2	.8

X_2	X_3	X_4 = t	X_4 = f
t	t	.9	.1
‘ t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$p(X_2 = x_2, X_4 = f) = \sum_{x_1, x_3 \in \{t,f\}} p(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = f)$$

$$= \sum_{x_1, x_3} \Big[p(X_1 = x_1) p(X_2 = x_2 \mid X_1 = x_1) p(X_3 = x_3 \mid X_1 = x_1)$$

$$p(X_4 = f \mid X_2 = x_2, X_3 = x_3) \Big]$$

$$= \ldots$$

Variable Elimination: Example

$$p(X_2 = x_2, X_4 = f)$$

$$= \sum_{x_1} p(X_1 = x_1) p(X_2 = x_2 \mid X_1 = x_1) \Big[\sum_{x_3} p(X_3 = x_3 \mid X_1 = x_1)$$

$$p(X_4 = f \mid X_2 = x_2, X_3 = x_3) \Big]$$

$$= \sum_{x_1} p(X_1 = x_1) p(X_2 = x_2 \mid X_1 = x_1) F_1(X_1 = x_1, X_2 = x_2) = F_2(X_2 = x_2)$$

where

	X_3	
X_1	*t*	*f*
t	.7	.3
f	.2	.8

		X_4	
X_2	X_3	*t*	*f*
t	*t*	.9	.1
t	*f*	.7	.3
f	*t*	.8	.2
f	*f*	.4	.6

$\mapsto$

x_1	x_2	$F_1(X_1, X_2)$
t	*t*	.7· .1 + .3· .3 = .16
t	*f*	.7·.2 + .3·.6 = .32
f	*t*	.2·.1 + .8·.3 = .26
f	*f*	.2·.2 + .8·.6 = .52

and

	X_1	
	t	*f*
	.5	.5

	X_2	
X_1	*t*	*f*
t	.7	.3
f	.1	.9

x_1	x_2	$F_1(X_1, X_2)$
t	*t*	.16
$\vdots$	$\vdots$	$\vdots$

$\mapsto$

x_2	$F_2(X_2)$
t	...
f	...

Complexity

- Variable elimination is exponential in maximal number of arguments of factors $p(\ldots \mid \ldots)$ resp. $F_j(\ldots)$ that appear in the summation process.
- This number depends strongly on the network structure
- ... and can also depend strongly on the order in which we sum out the variables!

Approximate inference using sampling

Problem Structure

Input: Evidence $\mathbf{X} = \mathbf{x}$, random variable Z, value $z \in dom(Z)$.
Output: Find approximation q for $p := P(Z = z \mid \mathbf{X} = \mathbf{x})$.

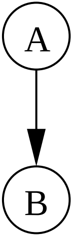
Grand plan

- Somehow sample instantiations from the domain $\mathbf{Y}$; $\mathbf{X} \cup Z \subseteq \mathbf{Y}$.
- Somehow use the samples $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ to find the approximation q.

Forward Sampling

Observation: can use Bayesian network as random generator that produces full instantiations $\mathbf{Y} = \mathbf{y}$ according to distribution $P(\mathbf{Y})$.

Example:



	A	
	t	f
	.2	.8

	B	
A	t	f
t	.7	.3
f	.4	.6

- Generate random numbers r_1, r_2 uniformly from [0,1].
- Set $A = t$ if $r_1 \leq .2$ and $A = f$ else.
- Depending on the value of A and r_2 set B to t or f.

Generation of one random instantiation: linear in size of network.

Sample Estimate

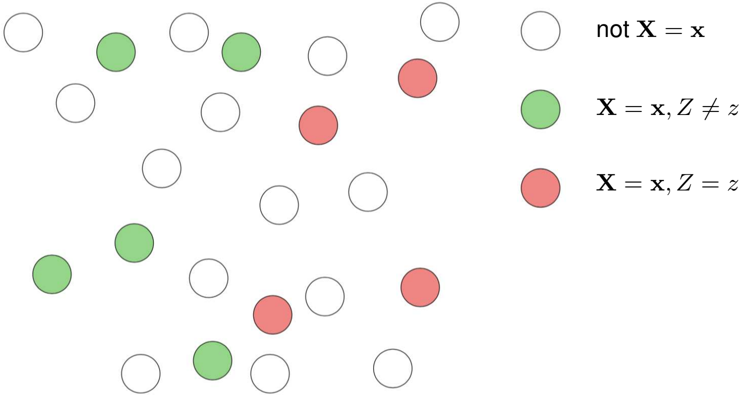
- Given a sample $\mathbf{y}_1, \ldots, \mathbf{y}_N$ of complete instantiations generated (independently) by the sampling algorithm, approximate $P(\mathbf{X} = \mathbf{x})$ as

$$q^* := \frac{1}{N} |\{i \in 1, \ldots, N \mid \mathbf{X} = \mathbf{x} \text{ in } \mathbf{y}_i\}|$$

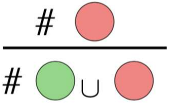
- Similarly, the sample provides an estimate for $P(Z = z, \mathbf{X} = \mathbf{x})$.
- Put together, we can estimate

$$P(Z = z \,|\, \mathbf{X} = \mathbf{x}) = P(Z = z, \mathbf{X} = \mathbf{x}) / P(\mathbf{X} = \mathbf{x}).$$

Forward Sampling: Illustration

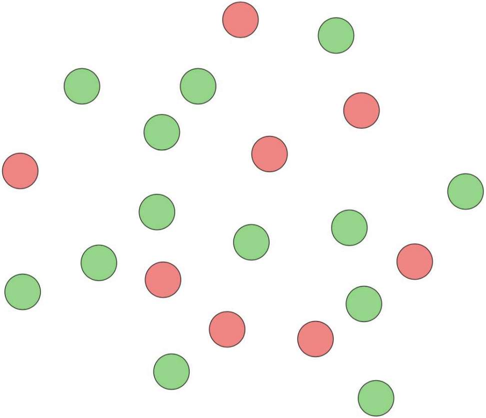


Approximation for $P(Z = z \mid \mathbf{X} = \mathbf{x})$:



Problem of forward sampling: samples with $\mathbf{X} \neq \mathbf{x}$ are useless!

Goal: find algorithm that samples according to $P(\mathbf{Z} \mid \mathbf{X} = \mathbf{z})$:



Approximate inference – Gibbs Sampling

- **Principle:** obtain new sample from previous sample by randomly changing the value of only one selected variable.
- **Notation:** Let $\mathbf{Y} = (\mathbf{Z}, \mathbf{X})$ denote all variables in the domain, where $\mathbf{X} = \mathbf{x}$ is observed.

Gibbs sampling

$\mathbf{z}_0$:= arbitrary instantiation of $\mathbf{Z}$.
$\mathbf{y}_0$:= $(\mathbf{z}_0, \mathbf{x})$.
t := 1.
repeat forever
    choose $Z_k \in \mathbf{Z}$
    set $y_{t,j} := y_{t-1,j}$ for all Y_j except the chosen Z_k.
    generate randomly $z_{t,k}$ according to $P\left(Z_k \,|\, \mathbf{Y} \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathbf{Y} \setminus \{Z_k\}}\right)$
    Store the sampled value in Z_k's location in $\mathbf{y}_t$.
    t := $t + 1$.

Resampling Z_k

$$
\begin{aligned}
P\big(Z_k = z_k \mid \mathbf{Y} \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathbf{Y} \setminus \{Z_k\}}\big) \\
&\propto P(Z_k = z_k, \mathbf{Y} \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathbf{Y} \setminus \{Z_k\}}) \\
&\propto P(Z_k = z_k \mid \mathrm{pa}(Z_k) = \mathbf{y}_t^{\downarrow \mathrm{pa}(Z_k)}) \cdot \\
&\qquad \prod_{i : Y_i \in \mathrm{ch}(Z_k)} P(Y_i = y_{t,i} \mid \mathrm{pa}(Y_i) \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathrm{pa}(Y_i) \setminus \{Z_k\}}, Z_k = z_k) \quad (*),
\end{aligned}
$$

where $\propto$ means: equals up to a constant that does not depend on z_k.

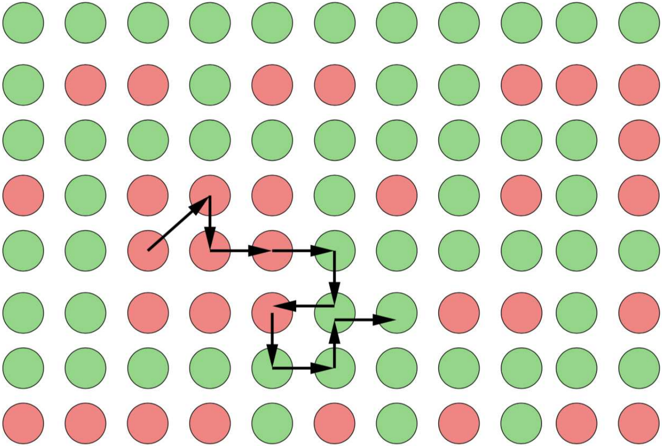
$\leadsto$ **Note:** To sample a value we only need to consider the **Markov blanket** for Z_k!

To sample $Z_{t,k}$

- Normalize $(*)$ to obtain $P\left(Z_k \mid \mathbf{Y} \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathbf{Y} \setminus \{Z_k\}}\right)$
  - $P\left(Z_k = z_k \mid \mathbf{Y} \setminus \{Z_k\} = \mathbf{y}_t^{\downarrow \mathbf{Y} \setminus \{Z_k\}}\right)$ is called the *full conditional* for Z_k.
- Sample value $z_{t,k}$ according to the resulting distribution

Gibbs Random Walk

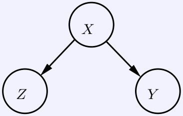
The process of Gibbs sampling can be understood as a *random walk* in the space of all instantiations with $\mathbf{X} = \mathbf{x}$:



Reachable in one step: instantiations that differ from current one by value assignment to at most one variable (assume randomized choice of variable Z_k).

Code Task: Exact and approximate inference

In this exercise you should implement a Gibbs sampler for the linear Gaussian model



where the distributions are given as

$$f(x) = \mathcal{N}(x|\mu_x, \sigma_x^2) \qquad f(y|x) = \mathcal{N}(y|x, \sigma^2) \qquad f(z|x) = \mathcal{N}(z|x, \sigma^2).$$

Start with the partial implementation

`students_gibbs_sampling.ipynb`.

Things to try out:

- How does changing the number of samples affect the accuracy of the approximation?
- Try experimenting with different values for the initial parameters.
- Are your results sensitive wrt. the starting position? Why (not)?