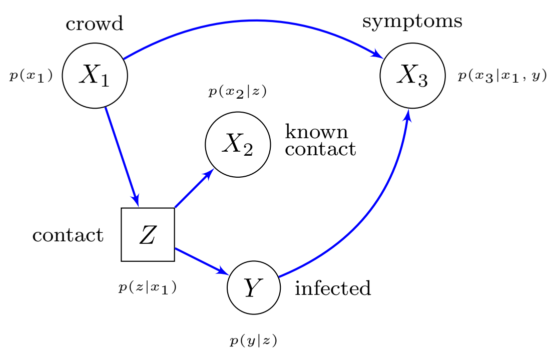
Extra Exercises for Models of Complex Systems

2022-03-03

Exercise 1. Consider the following Bayesian network over binary variables X_1, X_2, X_3, Y, Z.



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

Consider the following data, where in all tables $\hat{p} = \hat{p}(1|\ldots)$, the estimated conditional probability of a 1:

	y		
z	0	1	$\hat{p}$
0	9990	10	0.001
1	800	200	0.200

$p(y|z)$

	x_2		
z	0	1	$\hat{p}$
0	NN	0	0.0
1	50	50	0.5

$p(x_2|z)$

	x_3		
(x_1, y)	0	1	$\hat{p}$
(0,0)	980	20	0.02
(1,0)	950	50	0.05
(0,1)	20	80	0.80
(1,1)	15	85	0.85

$p(x_3|x_1, y)$

	z		
x_1	0	1	$\hat{p}$
0	990	10	0.01
1	190	10	0.05

$p(z|x_1)$

x_1		
0	1	$\hat{p}$
950	50	0.05

$p(x_1)$

Compute..

- ..the conditional probability $p(x_2|x_1)$ for $x_1 = 0, x_2 = 1$.
- ..the conditional probability $p(y|x_2, x_1)$ for $y = 1, x_1 = 1, x_2 = 0$.
- ..the probability $p(y, x_1, x_2, x_3)$ for $y = 0, x_1 = 0, x_2 = 1, x_3 = 1$.
- ..the conditional probability

$$p(y|x_1, x_2, x_3) = P(Y = y \mid X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

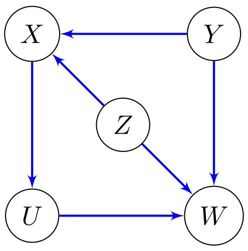
  for $y = 0, x_1 = 1, x_2 = 1, x_3 = 0$.

Exercise 2. How many directed graphs are I-equivalent to the directed chain $X_1 \to X_2 \to \ldots \to X_d$?

Exercise 3. Watch the first 4 minutes of the *Reachable* video available on the course page. On slide 11, Niels mentions that "it's probably a good idea to think in terms of the topological ordering of the nodes" and that the topological ordering may be useful for implementing the Reachable algorithm.

Discuss why the topological ordering is helpful in phase 1 of the Reachable algorithm. For example, provide an example of a Bayesian structure $\mathcal{G}$, nodes $\mathbf{Z}$ in $\mathcal{G}$, and a node $X \notin \mathbf{Z}$ and illustrate how, for your chosen setting, phase 1 benefits from traversing the nodes in the reverse topological ordering by contrasting it to another less suitable ordering.

Exercise 4. Determine which of the conditional independencies below hold for all distributions that factorize according to the Bayesian structure:


$$\begin{tikzcd} X \arrow[dd] & & Y \arrow[ll] \arrow[dd] \\ & Z \arrow[ul] \arrow[dr] & \\ U \arrow[rr] & & W \end{tikzcd}$$

- $Z \perp U \mid X$
- $Z \perp Y \mid U$
- $X \perp W \mid U, Z, Y$
- $Z \perp U \mid X, W$
- $Y \perp U \mid X, Z$

Exercise 6. Consider the following two specifications of the joint distribution of three random variables X, Y, Z with $\mathrm{Val}(X) = \mathrm{Val}(Y) = \mathbb{N}_0$ and $\mathrm{Val}(Z) = \{0, 1\}$.

- $P(Z = 1) = 0.5$, $X \perp Y \mid Z$, $P(X \mid Z = 0) = P(Y \mid Z = 0)$ is a Poisson distribution with mean 1 and $P(X \mid Z = 1) = P(Y \mid Z = 1)$ is a Poisson distribution with mean 5.
- $X \perp Y$, $P(X) = P(Y)$ is a Poisson distribution with mean 5 and
$$Z = \begin{cases} 1 & \text{if } |X - Y| \leqslant 2 \\ 0 & \text{otherwise} \end{cases}$$

Draw the graphs of the Bayesian networks that correspond to the distributions. Write a program that simulates from either distribution and simulate 1000 triples (x_i, y_i, z_i). Make three plots for both distributions: one plot of y_i against x_i, one plot of y_i against x_i for those i with $z_i = 0$ and one plot of y_i against x_i for those i with $z_i = 1$. Comment on the results.
Hint for plotting: When plotting many values for discrete variables there is a lot of overplotting, which prevents you from seeing patterns. Using transparency *(`alpha = 0.3` in matplotlib's `scatter` function), and* jittering *(use e.g. `regplot` from the seaborn library) can help resolve this issue.*

Exercise 8. Consider three discrete variables X, Y and U and assume that their joint distribution has point probabilities that factorize as
$$p(x, y, u) = \frac{1}{Z}\varphi_1(x, u)\varphi_2(y, u).$$
We know that this means $X \perp Y \mid U$ and that $p(x, y \mid u) = p(x|u)p(y|u)$. However, for this exercise we will investigate the computation of $p(x, y|u)$ directly from the factors φ_1 and φ_2. Let
$$\varphi^*(u) = \sum_{x,y} \varphi_1(x, u)\varphi_2(y, u),$$
let $n_1 = |\mathrm{Val}(X)|$ and let $n_2 = |\mathrm{Val}(Y)|$.

- Argue that
$$p(x, y|u) = \frac{\varphi_1(x, u)\varphi_2(y, u)}{\varphi^*(u)}.$$
(Hence, no need to compute Z.)
- Let us compute $\varphi^*(u)$ for a fixed u directly from the definition by first computing $\varphi_1(x, u)\varphi_2(y, u)$ and then summing out. Express the number of multiplications and additions needed in terms of n_1 and n_2.
- Let us then rewrite $\varphi^*(u)$ using variable elimination (the order of x and y does not matter). Using the rewritten form, express the number of multiplications and additions now needed in terms of n_1 and n_2.

Exercise 9.* Write a program that computes the partition function for the Ising model of the variables $(X_1, \ldots, X_n)$ with the Markov network being
$$X_1 — X_2 — X_3 — \ldots — X_{n-1} — X_n$$
with n variables. Let all interaction parameters be the same, that is,
$$\omega_{i,j} = \omega \in \mathbb{R},$$

for $|i-j|=1$ (and $\omega_{i,j}=0$ otherwise). Also, put $u_i=0$. Hence, you need to compute

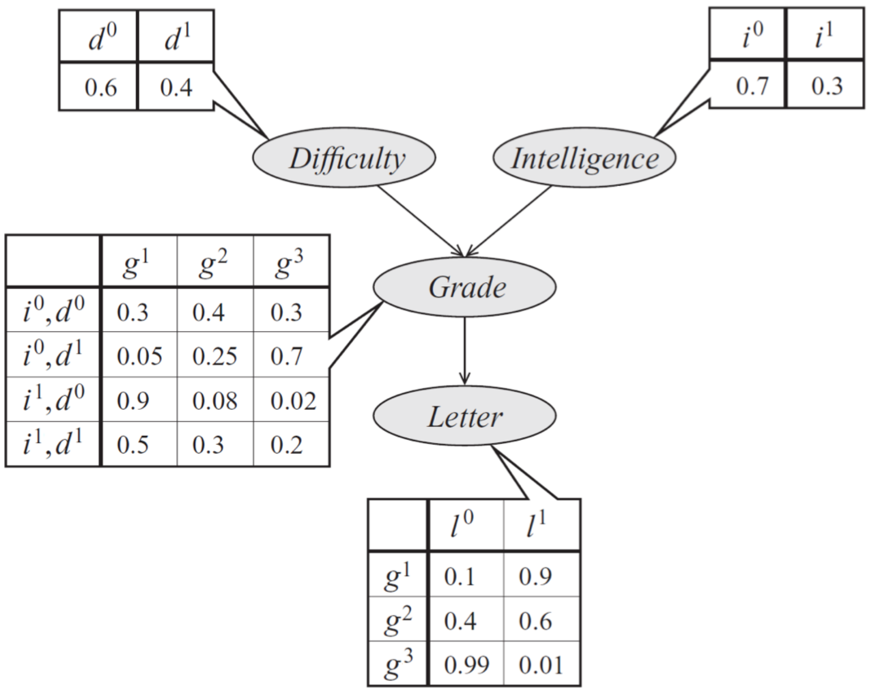
$$Z(\omega)=\sum_{(x_1,\ldots,x_n)\in\{-1,1\}^n}\exp\left(-\omega\sum_{i=1}^{n-1}x_ix_{i+1}\right).$$

Plot $Z(\omega)$ as a function of ω. How large an n can you get up to?
Hint: There are 2^n outcomes. The biggest challenge of a brute-force implementation is probably to systematically loop through all possible outcomes. A (much, much) more efficient implementation uses variable elimination.

Exercise 11. In this exercise we consider a Bayesian network structure called an *arborescence* or a *directed rooted tree*, for which there is a root node, X_0, such that there is a unique path from X_0 to all other nodes. The leaves, $\mathbf{E}$, are nodes without children.

- Argue that the root has no parents and all other nodes have one parent.
- How many immoralities are there?
- Describe the moral graph.
- Suppose evidence, $\mathbf{e}$, is given by observing the leaves. Find a useful elimination order for computing $P(X_0 \mid \mathbf{E}=\mathbf{e})$.

Exercise 12. Consider the following Bayesian network structure and (conditional) probability tables:



d^0	d^1
0.6	0.4

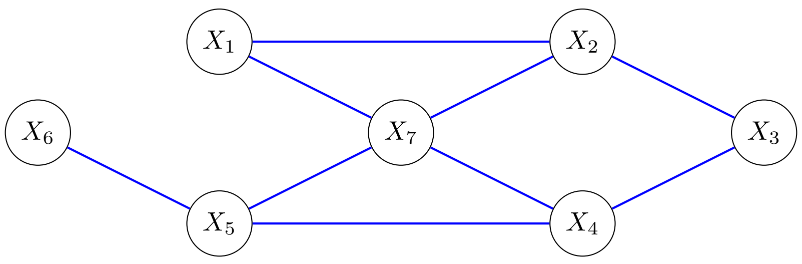
i^0	i^1
0.7	0.3

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

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

- Implement a function that samples one observation of the variable quadruple (Difficulty, Intelligence, Grade, Letter) and use it to simulate 100 observations of the provided Bayesian network.
- Based on the 100 simulated observations, estimate $P(\text{Grade}=g^1)$ and $P(\text{Grade}=g^2 \mid \text{Difficulty}=d^1)$.
- Now implement a routine that amounts to performing exact inference to obtain $P(\text{Grade}=g^1)$ and $P(\text{Grade}=g^2 \mid \text{Difficulty}=d^1)$.

Exercise 13. Consider the following graph



Consider first the elimination order $X_7, X_6, X_5, X_4, X_3, X_2, X_1$.

- Run the Sum-Product variable elimination algorithm on paper and determine the dimensions of all ψ- and τ-factors generated. Draw the induced graph.

- What is the largest dimension of ψ-factors generated, and how many factors of this dimension are needed?

Other elimination orders may be preferable. The greedy search algorithm, Algorithm 9.4, can be used to choose a better elimination order. In the question below you may break ties of the evaluation metric by choosing the node with the smallest index.

- Using the min-fill metric, find the elimination order produced by the greedy search algorithm.
- Run the Sum-Product variable elimination algorithm with the greedily selected order. Determine the dimensions of all ψ- and τ-factors generated and draw the induced graph. Comment on differences between this graph and the induced graph from the other elimination order.
- What is now the largest dimension of ψ-factors generated?

Exercise 14. In this exercise you first consider two factors

$$\varphi_1(x, y) = e^{xy} \quad \text{and} \quad \varphi_2(y, z) = e^{zy}.$$

- Write a Python function, `tau0`, which computes

  $$\tau_0(x, z) = \sum_y \varphi_1(x, y)\varphi_2(y, z).$$

  The Python function `tau0` needs to take two arguments, the x and the z. It has to sum out values of y, and you may assume at first that they are $\{+1, -1\}$. Test the function with e.g. `tau0(1, -1)` and `tau0(-1, -1)`.
- Write another Python function, `tau_factory`, that takes two arguments, both being Python functions, so that `tau1 = tau_factory(lambda x,y: np.exp(x * y), lambda y,z: np.exp(y * z))` gives a function, `tau1`, which does the same as `tau0`. It may be a good idea to give `tau_factory` a third argument with default value $\{+1, -1\}$, which is the vector of y-values to sum out. In that way you do not need to hardcode the values y takes into the function.

The function `tau_factory` is creating and returning a function, thus it is a *function factory*, hence the name. With such a function factory for creating factors from other factors by the Sum-Product construction, we can go ahead an recursively create factors. Consider the factors

$$\varphi_3(z, w) = e^{2zw} \quad \text{and} \quad \varphi_4(w, u) = e^{-wu}$$

together with

$$\tau_2(x, w) = \sum_z \tau_1(x, z)\varphi_3(z, w)$$
$$\tau_3(x, u) = \sum_w \tau_2(x, w)\varphi_4(w, u)$$

- Set `tau2 = tau_factory(tau1, lambda z,w: np.exp(2 * z * w))` and `tau3 = tau_factory(tau2, lambda w,u: np.exp(- w * u))` and test that `tau3` computes τ_3.
- Suppose that $n = |\mathrm{Val}(Y)| = |\mathrm{Val}(Z)| = |\mathrm{Val}(W)|$. How does the run time of, for example, the `tau3(-1, 1)` evaluation scale with n? Discuss how this scaling compares to an implementation that computes and stores $\tau_1(-1, z)$ as an array instead. If you want to benchmark run times in Python, you might want to choose values of the variables in $[-1, 1]$ to avoid overflow.

Exercise 16 (Auto-integration)**.** This exercise is similar to Exercise 14 except that we substitute the discrete valued sets with intervals and summation with intergration. Thus consider factors

$$\varphi_1(x, y) = e^{xy} \quad \text{and} \quad \varphi_2(y, z) = e^{zy}$$

for $x, y, z \in [-1, 1]$.

- Write a Python function, `tau_int`, which computes

  $$\tau(x, z) = \int_{-1}^{1} \varphi_1(x, y)\varphi_2(y, z)\ \mathrm{d}y.$$

  Test the function with e.g. `tau_int(1, -1)` and `tau_int(1, 1)`.
- Write another Python function, `tau_int_factory`, that takes two arguments, both being Python functions, so that `tau_int_1 = tau_int_factory(lambda x,y: np.exp(x * y), lambda y,z: np.exp(y * z))` gives a function, `tau_int_1`, which does the same as `tau_int`.
  Hint: While you might have implemented `tau_int` *by analytically working out what the formula for the integral is, you cannot do so here. One way to proceed is to use numerical integration. This can be done with the* `quad` *function from the* `scipy.integrate` *module. Thus your function could be implemented using something like the line*
  `quad(lambda y: phi1(x, y) * phi2(y, z), -1, 1)[0]`

- Discuss how the above construction would generalize to nested numerical integration in an Integrate-Product algorithm and how that would work. Then discuss the possibility of implementing a Python function that produces efficient code, avoiding numerical integration, for evaluating Integration-Products.

Comment: In machine learning, libraries or tools for automatic differentiation have become de-facto standards for deriving gradients and higher order derivatives directly from code. Inference in graphical models requires integration, and this exercise explores the possibility of automatic integration.

Exercise 17. The *Gamma distribution* on $(0, \infty)$ with *shape* parameter $\alpha > 0$ and *rate* parameter $\beta > 0$ has density

$$p(y) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} e^{-\beta y}$$

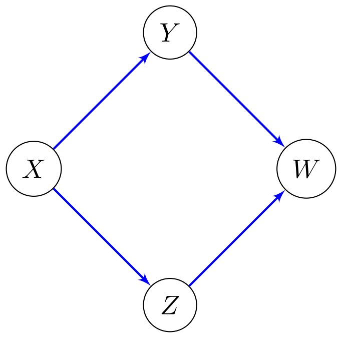
where

$$\Gamma(\alpha) = \int_0^\infty y^{\alpha-1} e^{-y} \, \mathrm{d}y$$

is the Γ-integral. In this exercise we suppose that X follows a Gamma distribution with shape α_0 and scale $\beta = 1$ and that $Y \mid X = x$ follows a Gamma distribution with shape α_1 and scale $\beta = x$.

- Find the density for the marginal distribution of Y.
- Show that the conditional distribution of X given $Y = y$ is a Gamma distribution with shape $\alpha_0 + \alpha_1$ and scale $1 + y$.
- How is the *conditional expectation* defined in the continuous case, cf. page 32 in the book? Compute $E(Y \mid X = x)$ and $E(X \mid Y = y)$.

Exercise 18. Consider the Bayesian network over


$$\begin{tikzcd} & Y \arrow[dr] & \\ X \arrow[ur] \arrow[dr] & & W \\ & Z \arrow[ur] & \end{tikzcd}$$

and suppose that all four CPDs are *linear Gaussian*, cf. Definition 5.14.

- Implement a Python function that will simulates (X, Y, Z, W) for any choice of parameters in the CPDs.
- Let $X \sim \mathcal{N}(0, 1)$, $Y \mid X = x \sim \mathcal{N}(x, 1)$, $Z \mid X = x \sim \mathcal{N}(x, 1)$ and $W \mid Y = y, Z = z \sim \mathcal{N}(z + y, 1)$, simulate 10.000 data points (x_i, y_i, z_i, w_i) and plot the histograms of all the marginal distributions.
- Plot w_i against x_i and carry out a linear regression of w_i on x_i. That is, estimate parameters $\hat{a}$ and $\hat{b}$ by least squares so that

  $$\hat{a} + \hat{b} x_i$$

  is the best (linear) approximation of w_i. Discuss the result, in particular if you could have computed upfront what these parameters would (approximately) be.

Exercise 20. A Gaussian Bayesian network with variables $X_1, \ldots, X_n$ has a density that factorizes as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa}_i) = \frac{1}{\sqrt{(2\pi)^n \sigma_1^2 \cdots \sigma_n^2}} e^{-\sum_{i=1}^n \frac{1}{2\sigma_i^2}(x_i - \beta_{i,0} - \beta_i^T \mathbf{pa}_i)^2}$$

where $\mathbf{pa}_i = (x_{j_1}, \ldots, x_{j_{k_i}})$ are the k_i parents of node i and

$$\beta_i^T = (\beta_{i,1}, \ldots, \beta_{i,k_i})$$

is the conditional mean parameter vector for the i-th node.

- Rewrite $p(\mathbf{x})$ to be on canonical form

  $$p(\mathbf{x}) = C(\mathbf{x}; J, \mathbf{h}, g) = e^{-\frac{1}{2}\mathbf{x}^T J \mathbf{x} + \mathbf{h}^T \mathbf{x} + g}$$

  and express the entries of J and $\mathbf{h}$ in terms of the β- and σ^2-parameters.

- Find also an expression for g and use this to derive a formula for $\det(J)$.
- Suppose that there is an $\mathbf{x} \neq 0$ so that $\mathbf{x}^T J \mathbf{x} \leqslant 0$. Argue via the canonical form that $p(z\mathbf{x})$ could then become arbitrarily large by either $z \to \infty$ or $z \to -\infty$, and that this would contradict that p is upper bounded. (The argument for $\mathbf{x}^T J \mathbf{x} < 0$ is easiest). Use this to conclude that J is positive definite.

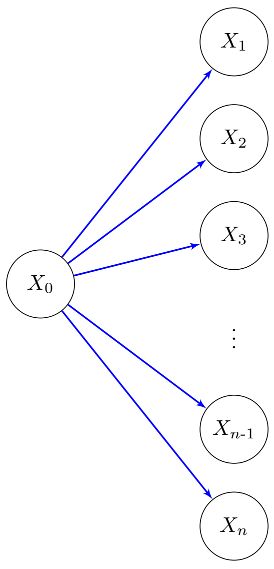
Exercise 21. Consider the five update equations

$$\begin{aligned} X_0 &= Z_0 \\ X_1 &= 2 + 4X_0 + Z_1 \\ X_2 &= 1 + 2X_0 + Z_2 \\ X_3 &= Z_3 \\ X_4 &= X_1 + X_2 + X_3 + Z_4 \end{aligned}$$

where Z_0, Z_1, Z_2, Z_3, Z_4 are independent and $\mathcal{N}(0,1)$-distributed.

- Derive the information parameters J and $\mathbf{h}$ for the joint Gaussian distribution of X_0, X_1, X_2, X_3, X_4
- Draw both the Bayesian network structure and the Markov network structure corresponding to J. How are these graphs related?
- Find the conditional distribution of (X_0, X_3) given (X_1, X_2, X_4)

Exercise 22. In this exercise we consider the Gaussian Bayesian network with the following graph



Thus $P(X_0) = \mathcal{N}(\beta_{0,0}, \sigma_0^2)$ and

$$P(X_i \mid X_0 = x) = \mathcal{N}(\beta_{i,0} + \beta_{i,1}x; \sigma_i^2)$$

for parameters $\beta_{0,0}, \beta_{1,0}, \ldots, \beta_{n,0}, \beta_{1,1}, \ldots, \beta_{n,1} \in \mathbb{R}$ and $\sigma_0^2, \sigma_1^2, \ldots, \sigma_n^2 > 0$.

- Implement a function that for given parameters simulates N vectors of length n each from this model.
- Implement another function that computes the information parameters J and $\mathbf{h}$ in terms of the β- and σ^2-parameters.
- Implement a function that uses the Theorem 7.4 formulas to compute the conditional distribution of $X_0 \mid X_1, \ldots, X_n$.
- Implement another function that uses the information parameters directly to compute the conditional distribution of $X_0 \mid X_1, \ldots, X_n$.
- Compare the results you get to the results of a multiple linear regression of X_0 on $X_1, \ldots, X_n$ from the simulated data.

You may experiment with different choices of n and parameters. Choose a small n like $n = 5$ at first. To easily get a variety of parameters use simulations to also generate the parameters, e.g. Gaussian β-parameters and exponentially or gamma distributed σ^2 parameters.