Complex-Network Modelling and Inference

Lecture 3: Application: Bayesian Networks (and Complexity)

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
https://roughan.info/notes/Network_Modelling/

School of Mathematical Sciences,
University of Adelaide

March 7, 2024

Section 1

Big-O Notation (and its friends)

Computational complexity

- Often, we don't care about the time for a particular problem, we care about the practical bounds for problems we might consider in the future
- We would like to estimate how long our program will take to run
  - as a function of the *size* of the problem
    - *e.g.*, n equals the number of variables
    - *e.g.*, m equals the number of constraints
  - could also include the size of the variables in memory
    - *e.g.*, k bit floating point numbers
- often interested in BIG problems, so look at asymptotic behaviour
  - *e.g.*, large m and n
  - use big-O notation

Big-O notation

Definition

$$f(\mathrm{x}) = O\big(g(\mathrm{x})\big)$$

means (i.e., iff) there exists constant c and x_0 such that

$$|f(\mathrm{x})| \leq c|g(\mathrm{x})|$$

for all x *such that* $x_i \geq x_0$.

Usage:

- describes asymptotic limiting behaviour: implicit that $x \to \infty$
- the function $g(x)$ is chosen to be as simple as possible
- a common *mistake* is to think that it means $f(x)/g(x) \to k$

Big-O notation properties

- Multiplication: $f_1 = O(g_1)$ and $f_2 = O(g_2)$ then

$$f_1 \times f_2 = O(g_1 \times g_2)$$

- Multiplication by a constant: $f = O(g)$

$$kf = O(g)$$

- Summation: $f_1 = O(g_1)$ and $f_2 = O(g_2)$ then we can write a general expression, but usually either $g_1 = g_2$, or WLOG g_1 grows faster than g_2 and in these cases

$$f_1 + f_2 = O(g_1)$$

These properties mean that we can simplify using a simple set of rules

Big-O notation rules

When we use Big-O notation, we use the following rules:

1. if $f(x)$ is a sum drop everything except the term with the largest growth rate
2. if $f(x)$ is a product any constants are ignored

Assume these rules have been applied, when you see Big-O.

Example of RULE-1

Example

$f(x) = x^7 - 200x^4 + 10$ is dominated (for large x) by the x^7 term, so

$$f(x) = O\left(x^7\right)$$

We dropped the terms $-200x^4 + 10$ because they grow slower than x^7.

Example

We can reduce $O(n^2 + \log n)$ to $O(n^2)$.

The $\log(\cdot)$ function grows more slowly than n (or any polynomial).

Example of RULE-2

Example

$f(n) = 3n^2$, which is a product, so we ignore constants, and

$$f(n) = O(n^2)$$

We ignored the constant 3.

Example

If k is a constant, we can rewrite $O(kn \log n)$ as $O(n \log n)$.

Whether k is a constant depends on the context.

Stirling's approximation

Stirling's approximation is both an example of use of the notation, and also a useful tool in some analysis:

$$\ln n! = n \ln n - n + O(\ln n)$$

We use Big-O notation here

We will use Big-O notation to count operations in an algorithm

Classic examples

problem	complexity	notes
$\sum_{i=1}^{n} x_i$	$O(n)$	
$A \times B$	$O(n^3)$	naïve algorithm
	$O(n^{2.373})$	clever algorithm
A^{-1}	$O(n^3)$	naïve algorithm
	$O(n^{2.373})$	clever algorithm
$det(A)$	$O(n!)$	naïve algorithm
	$O(n^3)$	clever algorithm

Where A and B are $n \times n$ matrices

Example of a more complicated function

Example

Calculate the complexity of computing $f(x) = \exp(x)$.

- This depends on how you compute $\exp(x)$.
- A simple approach is Taylor series
  - assume you want n digits of precision
  - that determines how many terms you need in the Taylor series
  - so computation is $O(nM(n))$, where $M(n)$ is the cost of a multiplication with n digits
- Assuming fixed precision (*e.g.*, in MATLAB, double precision)

$$\exp(x) = O(1)$$

  That is, its computational time doesn't depend on how big x is
- There are faster approaches, but this suffices for today
- Other elementary functions, *e.g.*, sin, cos, arctan, log, are similar

Nomenclature

In order, we describe classes of algorithms as ???-time (*e.g.*, constant-time)

complexity	name	example algorithms
$O(1)$	constant	calculate simple functions
$O(\log n)$	logarithmic	binary search
$O(n)$	linear	adding arrays of length n
$O(n \log n)$	log linear	Fast Fourier Transform (FFT)
$O(n^2)$	quadratic	adding up all elements of a matrix
$O(n^d)$	polynomial	naïve matrix multiplication
$O(c^n)$	exponential	Simplex
$O(n!)$	factorial	brute force search for TSP

Weirdness

Example

$$x = O(x^2) \quad \text{but} \quad x^2 \neq O(x)$$

so using $=$ is slightly weird, as there is an asymmetry.
Sometimes we use $\in$ instead.
e.g.,

$$x \in O(x^2)$$

Often the symbols are used more generally

Sometimes we use these symbols in a type of algebra

Example

$$\left(n + O(n^{1/2})\right)\left(n + O(\log n)\right)^2 = n^3 + O(n^{5/2})$$

Meaning: for any functions which satisfy each $O(\ldots)$ on the LHS, there are some functions satisfying each $O(\ldots)$ on the RHS, such that substituting all these functions into the equation makes the two sides equal.

Variables

It can get confusing, as variables and constants sometimes are inferred from context.

For instance

$$
\begin{aligned}
f(n) &= O(n^m) \\
g(m) &= O(n^m)
\end{aligned}
$$

mean quite different things, even though the RHSs are the same.

Big-O limitations

Big-O has advantages:

- it gets to the nub of the question – what is the *shape* of the performance of our algorithm for large problems

However it has limitations

- it doesn't tell us about constants, and lower-order terms
  - these are important, particular for small to moderate sized problems
  - Big-O is only for asymptotic performance
- it doesn't tell us actual computation times
- *it's only an upper bound*

Big-Ω

Two forms of Big-Omega notation

- Hardy-Littlewood (used in math)
- Knuth (used in computational complexity)

Definition (Big Omega)

$$f(x) = \Omega\big(g(x)\big) \Leftrightarrow g(x) = O\big(f(x)\big)$$

More succinctly: $f(x) \geq kg(x)$ for some k

- Similar to Big-O, but gives a lower bound

Big Theta

Definition (Big Theta)

$$f(x) = \Theta\big(g(x)\big)$$

means that $f(\cdot)$ is bounded above and below by $g(\cdot)$, *i.e.*,

$$k_1 g(x) \leq f(x) \leq k_2 g(x)$$

for positive constants k_1 and k_2, for all $x > x_0$.

So Big-Θ notation means the function $f(x)$ grows as fast as $g(x)$.

Section 2

Bayesian Networks

Graphs and Stats

- Statistics can be used to
  - analyse graphs/networks
  - estimate/infer graph properties
  - determine ways to sample from graphs
- Graphs can be useful in statistics
  - *graphical* structure can be used in models
  - *e.g.*, Bayesian networks

Multinomial Bayesian Network Example [SD15]

We want to model the relationships between a set of variables

Abb.	Variable	Values	Type
A	Age	{ young, adult, old }	demographic
S	Sex	{ M, F }	demographic
E	Education	{ high-school, uni }	socioeconomic
O	Occupation	{ self-employed, employee }	socioeconomic
R	Residence	{ small, large }	socioeconomic
T	Travel mode	{ car, train, other }	target variable

Goals

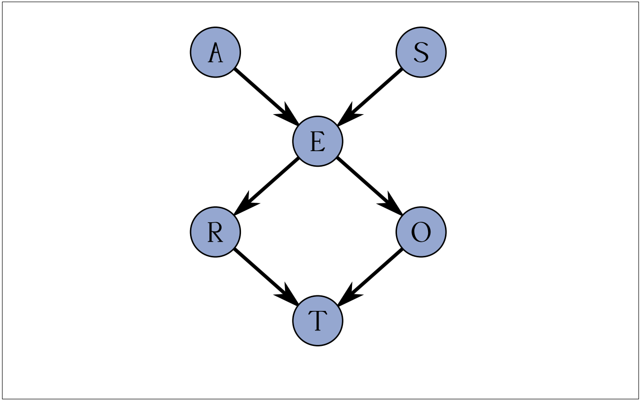
We could simply consider all possible associations but there is a large number of states, and hence parameters we would have to estimate. We want a way to simplify it.

Tasks of interest

1. Queries: given a set of relationships, derive the probability that a given person (of Age A, Sex S, ...) will they use a car to get to work, or given they drove, what are the probabilities of the other variables?
2. Estimation: given we are told which variables are directly related, find out the details of the relationships.
3. Identification: work out which variables are directly related.

Multinomial Bayesian Network Example [SD15]

Assume (for the moment) that we know the relationships between variables, the direct relationships can be shown as a DAG (a Directed Acyclic Graph), *e.g.*,



Called a *Bayesian Network* or a *Graphical Model*

DAG notation

Definition

Given a link or arc (i, j) in a DAG, the node i is called the *parent* of j, and the node j is called a *child* of i.

Parents and children only make proper sense when there are no cycles, otherwise, node A could be the parent of B, who is the parent of C, who is the parent of A, which isn't what I mean by parents.

Conditional Probability

- Each node/variable is associated with a conditional probability of the node's variable, conditional on its parents
- *e.g.*, Node R is associated with $P(R|E)$
- For categorical variables, it can be expressed in a table, *e.g.*,

E(ducation)	Pr{R=small}	Pr{R=large}
high-school	0.4	0.6
uni.	0.7	0.3

Multinomial Bayesian Network Example [SD15]

- A link means the variables are directly related, *i.e.*, that they are dependent, so we might say E *depends* on A
- There is an indirect relationship where-ever we can follow a path, *e.g.*, from A to T
- Absence of a link means the variables are conditionally independent, *e.g.*, there is no link from $E \to T$
  - This says that T and E are conditionally independent, given R and O (the nodes on the path between them), *i.e.*,

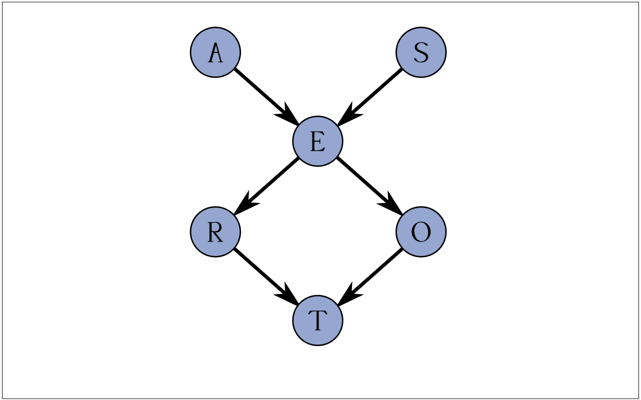
    $$Pr(T, E|O, R) = Pr(T|O, R)Pr(E|O, R)$$

    Further, as the arrows point from E to O and R, we say E doesn't depend on these variables, and so can reduce this further to

    $$Pr(T, E|O, R) = Pr(T|O, R)Pr(E)$$

Multinomial Bayesian Network Example [SD15]

Given this representation



We can write

$$Pr(A, S, E, O, R, T) = Pr(A)Pr(S)Pr(E|A, S)Pr(O|E)Pr(R|E)Pr(T|O, R)$$

Generally

- Nodes represent random variables
- Links represent dependence
- Can decompose joint probability distributions into a product of conditional probabilities of the form

$$Pr(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} Pr\big(X_i | parents(X_i)\big)$$

Multinomial Bayesian Network Example [SD15]

$$Pr(A, S, E, O, R, T) = Pr(A)Pr(S)Pr(E|A, S)Pr(O|E)Pr(R|E)Pr(T|O, R)$$

- This is a useful factorisation
  - reduces the number of parameters to model or estimate
    - estimation decomposed into small problems, *e.g.,* estimate $Pr(O|E)$
  - reduces calculations (not so hard here, but imagine a bigger example)
  - defines a nested set of models
- But where did I get the DAG in the first place?

Identifying DAG structure

Two main approaches

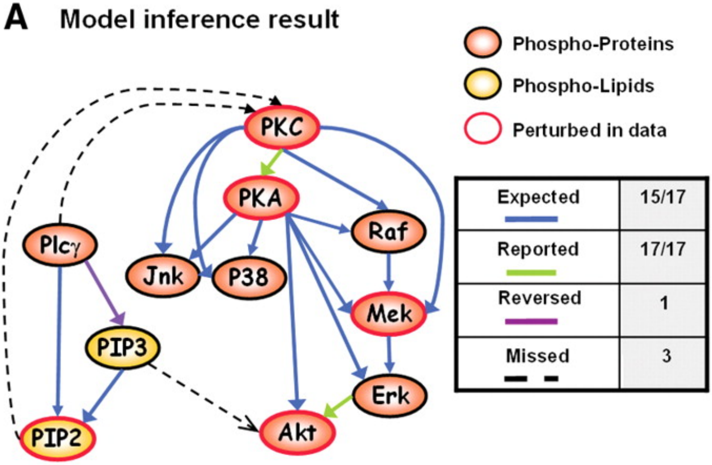
- link-by-link: consider each link separately
  - test for conditional independence between each pair of variables
  - but lots of tests, and conditions depend on other links
  - best for verifying an existing model
    - test if we should add or remove a link
    - *e.g.*, should there be a link $E \to T$?
- network-wide: given each network a "score", and choose the highest score
  - scores based purely on likelihood will prefer networks with lots of links
  - so use BIC (Bayesian Information Criteria)
  - but there are *very* many networks, so can't hope to score them all
  - lots of tricks and techniques
  - apparently can solve networks with ~ 100 nodes

A Real Example [SPP+05]

- Extra-cellular cues trigger a cascade of reactions
  - signalling molecules are activated
  - affect subsequent molecules
  - results in phenotype cellular response

  We would like to map these signalling pathways
- Traditional approach relies on diverse experiments, intuition, and lots of work
  - but can't consider pathways independently
  - cross-talk, and other complexities
- Intracellular multicolour flow cytometry provides a window into multiple signalling molecules.
  - simultaneous measures of protein expression levels
  - large sample sets
  - need a way to analyse the data

A Real Example [SPP+05]



Limitations

- This is a pretty small network
- Direction is not always meaningful
  - it implies a relationship
  - not necessarily causal
- Couldn't find all paths
  - Bayesian networks are inherently *acyclic*
  - Cell signalling pathways are not

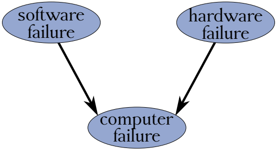
They had the advantage that this network was known before hand. Would I trust it if I didn't?

Reasoning with Bayesian Networks

- *Top-down*: causal reasoning
  - start with fixed state, or probabilities at the top
  - work out posterior probabilities of targets
- *Bottom-up*: explanatory or diagnostic reasoning
  - start with a state at the bottom, *e.g.*, catch the train
  - work out the likely cause

Used to build expert systems.

Weirdness in diagnostic reasoning



- Two causes "compete" to explain the failure
  - the two causes are independent, but
  - causes become conditionally dependent given common child
  - *e.g.*, if the computer has failed, and we know the hardware has failed, then the posterior probability of a simultaneous software failure goes down
- Call "explaining away"
- Its an example of Berkson's paradox
  - classic example: you are admitted to college for being "brainy" or "sporty"
  - one of these alone explains you being in the college
  - so the other becomes less likely

Other Graphical Models

Bayesian Networks are one member of the class of Graphical Models

- Models that are represented by a graph/network
- *e.g.*, special cases of BNs
  - Hidden Markov Models
  - Neural Networks
- *e.g.*, generalisations
  - Markov Random Fields (undirected edges)
  - dynamic BNs

Examples/Applications

- AT&T system to use customer data to work out which customers are likely to default on their bill
  - lots of other fraud detection, *e.g.*, credit cards
- Speech recognition
- Gene regulatory networks
- Image processing
- Sports betting

Further reading I

Marco Scutari and Jean-Baptiste Denis, *Bayesian networks with examples in R*, CRC Press, 2015.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan, *Causal protein-signaling networks derived from multiparameter single-cell data*, Science **308** (2005), no. 5721, 523–529.