Bayesian Network Visualization for Human-Machine Collaboration

Eric Lawton

May 6, 2016

Contents

Chapter 1

Introduction

1.1 Why this may be useful to you

I am learning how to apply machine learning to complex problems while making it easy for humans to follow the reasoning. I find network diagrams of all kinds a useful tool, so I'm exploring a particular kind, Bayesian Networks (BNs), for this purpose.

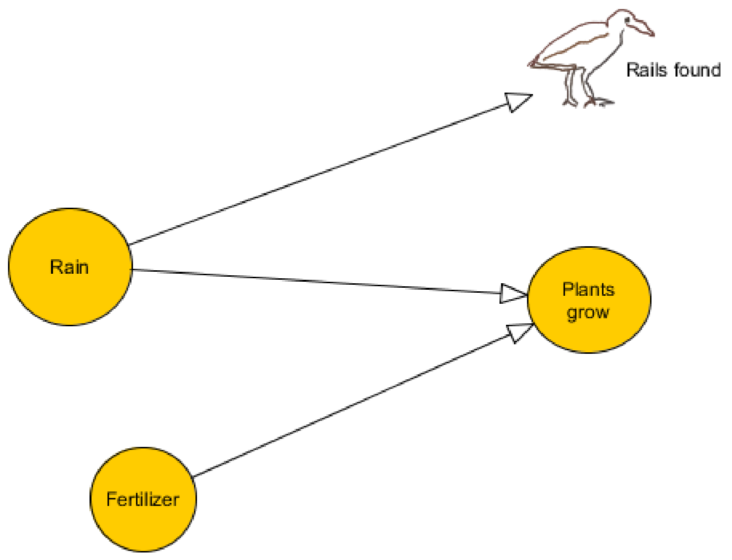
I'm currently interested in how we can use evidence to inform public policy-making, by governments and NGOs. However, so far it applies to any sets of related concepts (variables) where you may be interested in exploring correlations and causal relations, with questions such as:

- What is the structure of the relationships between these variables?
- Given a relationship, how strong an influence does one variable have on another?
- Given a variable with multiple causes, how do changes in the causes affect that variable?
- Where would more data help most with answering key questions?

I am documenting the resources I found most useful and also some blind alleys that I think I spent too much time on, so even though I'm still learning, this may save you some time if you are just starting.

Since the source learning material I am using has examples in the R programming environment, I'm using that for now. I also wanted to be able to publish working examples on the Web as I explore public data and it turns out that R offers Shiny to do just that. So far, it has worked flawlessly for me.

Figure 1.1: Example network



Since most interesting things that we want to understand, or to change, depend on many other things, it is often useful to draw diagrams to help with that understanding. Network diagrams are a helpful tool.

In the example above, there are four variables. Two of them are independent, for the purposes of this example; the rain and the fertilizer. How much my plants grow depend on these. [1]

The diagram shows these dependencies, with the arrows showing the direction of the dependencies. However, it soon gets more complex as more variables are introduced and we want to introduce statistical reasoning based on the evidence that supports these connections. The point of my investigation is how to use these relatively simple diagrams together with some machine learning algorithms, so that both humans and computers can collaborate in understanding such complex networks of causal connections.

1.2 Context and status

This is an early start on a work-in-progress. So far I have more questions than answers but I'll update it as I learn more. I just got to the point where the pages of obscure algebra are beginning to make sense, so I thought I'd get it captured at that stage before going further.

At the moment, I am doing all my exploring in the R programming environment. I tried Python but gave up for now as I need to focus a bit more. I will return later. See 4.1 for more details.

To follow this, you will need

- Some knowledge of programming in R, so you can play with the code. However I learned enough R to do this much in a few days (based on decades with other languages) so you may be able to at least read this with just general programming knowledge. Caution: R is somewhat quirky. All the code and this document is available on GitHub at https://github.com/EricLawtonCA/BayesNet.
  - For BN functionality, I am starting with is an R package called `bnlearn`[1] both because it has a good set of functionality and because there is a fairly good book to learn Bayesian Networks from, co-written by the author of the package.
  - For visualization, there is a choice. The main package I am using is `visNetworks`[2], as it is interactive, which I need in order to have the machine/human partnership. But so far I have not got it to display a static version in this document so you will see only screen shots of it here.

[1]So does the population of rails: a tip-of-the-hat to @RallidaeRule on Twitter™ as her feed introduced me to R. However, I have no data for them because they are elusive marsh birds.

  - A demonstration application is on Shiny.io at https://ericlawton.shinyapps.io/playShiny/

- ~~Some~~ No mathematics (yet). I took out the few equations I had, because it would take me too long to explain from scratch and they didn't add much. For next version, they'll come back in, so you'll need introductory undergraduate-level probability and statistics. I'd forgotten most of mine except for the bits I've used in practice, so I had to do a refresher. That was originally the main reason I learned all this R, so I could visualize the maths more easily. It worked, but now I need to think how to write it down in a simpler fashion than just replicating the text books. See 4.3 on page 14 for the principal sources I used.

If you have any feedback for me on this, please use Twitter. @Eric0Lawton (The zero is because I don't have a middle name and plain EricLawton was taken).

1.3 Vocabulary

Graph. Mathematicians reserve "graph" to mean objects that put nodes and edges together, as in the diagrams in this document. However, the term is used much more generally among non-mathematicians, to include many other kinds of plots, so I will use the work "network" instead, for clarity.

Node. I will stick to this instead of "vertex" because I don't know enough Greek to remember the plural "vertices". The same term is used by both `bnlearn` and `visNetwork`.

Edge. There is a slight problem here because `bnlearn` uses the graphical term "Arc", while `visNetwork` uses "Edge". This is one place where R is less helpful. When mapping data frames from one package to the other, you have to rename elements of the data structures. This feature is particularly bad for people who don't speak English. The names are hard-wired into the packages.

Parent. The node at the tail of an arrow is called a parent of the node at the pointy-end.

Child. The node at the head is called a child of its parent.

Chapter 2

Bayesian Networks

2.1 Summary

So far, I have learned the basics of BNs with discrete variables, and have assembled some machinery to make it all work in Shiny with `visNetwork`. The machinery has some basic selection and arrangement interaction but not yet an ability to restructure the graph or change the probability values. However, it does do the mapping from `bnlearn` to `visNetwork` and displays the associated CPTs when a node is selected.

The `bnlearn` classes do not keep track of visual elements and `visNetwork` doesn't keep track of probability tables, and they structure the data differently. So my first step was to find a way to transform the structures. They are relatively simple in that both use lists of nodes and of arcs that define which node they are "from" and which they are "to". The main difference is that `bnlearn` uses strings to identify nodes and `visNetwork` uses numeric IDs. So these need mapping back and forth. This is easy in R because both use named columns so it's just a question of adding a column and doing a lookup to map names to ids and back. The only other mapping I have implemented yet is from the strength of a connection to the width of an arc. So, for example, if rain has a bigger influence on plant growth than fertilizer, the arc from rain will be thicker than that from fertilizer. It doesn't make sense to map that back again as it is a calculated value that would not be set directly through the user interface.

2.2 Discrete Case: Multinomial Bayesian Networks

2.2.1 Summary of the theory

A BN is a Directed Acyclic Graph (DAG) where each node has a Conditional Probability Table (CPT) associated with it, also called a Node Probability Table.

"Directed" means there are arrows. Arcs have a direction from a variable to one that depends on it.

"Acyclic" means there are no closed loops. You can't follow a set of arrows from a node to another and finish up back where you started.

The nodes of the network represents the objects of interest (variables)

CPT is where Bayes comes in. Bayes Theorem is the mathematics of conditional probability.

For example, we may know from experience that the overall probability that plants grow a lot is 30%. However, if we know already that it rained but and they were fertilized, the probability will be even higher. For each combination of rain/not rain, fertilise/not fertilise, there is a different probability of the growth. This is called probability of growth, conditional on rain and fertilizer, written $P(Growth \mid Rain, Fertilize)$ and given in terms of a table, as in the examples below. A BN associates one of these tables with each node in the graph, conditional on the parents (the nodes "up arrow" from the node in question). If there are no parents, the CPT is just the probability distribution of the node. For example, we can get "Probability of precipitation" from the weather forecast.

Table 2.1: Rain CPT

Rained	Didn't rain
30	70

Table 2.2: Fertilizer CPT

Fertilized	Not fertilized
10	90

So for the example above, if we assume the discrete values of “rained yesterday” and “didn't rain yesterday” for “Rain” (as opposed to continuous value such as millimetres of rain), and other discrete values as shown in the following tables, the nodes would have the CPTs like those below, probabilities in percentages. (Omitting the rails because they are so elusive, I don't have any data).

There are various algorithms provided with BN software which allow the structure of the network and the CPTs to be provided by a human expert, or some or all of the structures and NPTs to be derived by algorithms from data sets. Sometimes only a partially complete structure or CPT set can be derived, where not enough data is provided, but it can also be seen where more data is needed either to estimate these or where the strength of the given estimates is weak. My next job is to learn more about these.

2.2.2 More Complex Example, with code

The book I am using ([3]) introduces an example concerning the use of different forms of transport by persons belonging to various sociological groups and provides a sample dataset containing the following variables, each of which has a very small set of discrete values.

Age: Age of person, divided into three classes; Young ($<$ 30 years), Adult (30 to 60) and Old ($>$ 60)

Sex: M or F (I know, a rather limited subset of possible values).

Education: high(-school), uni(versity)

Occupation: “Emp”(loyee) and “self”(-employed)

Residence: Size of the community in which the person lives, Small or Large

Transport: Most frequently used form, Auto, Train or Other

For our purposes, we can group these into three categories

Demographic: Age and Sex. Attributes of the person that cannot be modified

Socio-economic: Education, Profession and Residence, positions taken by people within society

Target-Variables: Transport. The variable which is the target of the study, of which this example has only one.

The following code fragments will read a dataset and automatically learn the structure that best fits the data. The network is then transformed into `visNetwork` format for display. For now, the dataset is an example I

Table 2.3: Plants grow CPT

		Grew a lot	**a bit**	**Didn't grow**
Fertilized	Rained	70	20	10
Not fertilized	Rained	50	30	20
Fertilized	Didn't rain	40	30	30
Not fertilized	Didn't rain	10	20	70

took from the bnlearn web site, so that I can work through the exercises to learn the network theory and also build the interactive machinery.

The code below will set things up by

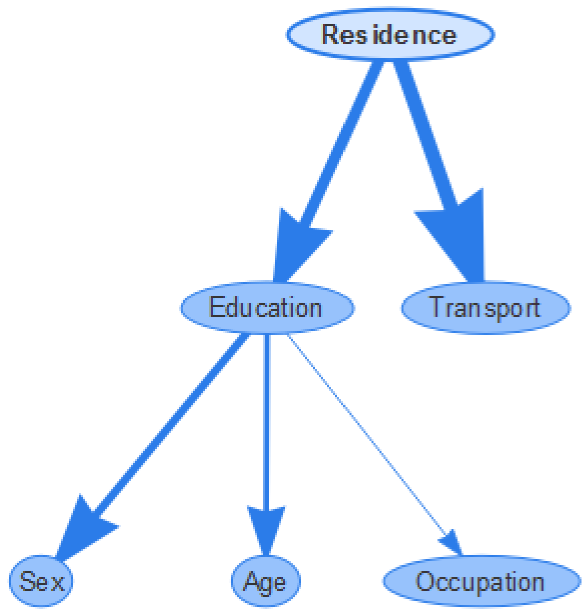
1. Reading the data
2. Getting the network structure, i.e. figuring out which arcs are needed to show the dependencies between the variables, and which variables should NOT be connected by arcs. This could be done by a human expert, but in this case I use a hill-climbing algorithm provided by `bnlearn`'s `hc` function. to analyze the data and return the basic DAG.
3. Calculating the strengths of those dependencies. In this case I use a χ^2conditional dependence test supplied with `bnlearn`. This function returns a p-value, the lower the number, the stronger the connection, so I invert that using a normalised reciprocal to make the arc thicker, the lower the number. It's just a rough approximation that I may need to refine later.
4. Calculating the CPTs for all the nodes, using `bn.fit`.

In one way, it would be convenient to do this all in one function call, but the advantage of doing it piecemeal is that you can inject human expertise instead, at each step, by specifying the data instead of calculating it. (Of course, the expert would probably calculate herself, but using different tools).

```
survey <- read.table("../data/survey.txt", header = TRUE)
dag <- hc(survey)
dag

##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [Residence][Education|Residence][Transport|Residence][Age|Education]
##    [Occupation|Education][Sex|Education]
##   nodes:                                 6
##   arcs:                                  5
##     undirected arcs:                     0
##     directed arcs:                       5
##   average markov blanket size:           1.67
##   average neighbourhood size:            1.67
##   average branching factor:              0.83
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 BIC (disc.)
##   penalization coefficient:              3.107304
##   tests used in the learning procedure:  40
##   optimized:                             TRUE

arc.strengths <- arc.strength(dag, data = survey, criterion = "x2")
arc.strengths

##        from         to     strength
## 1 Residence  Education 0.0005599201
## 2 Education        Sex 0.0005877751
## 3 Residence  Transport 0.0004095347
## 4 Education        Age 0.0008689951
## 5 Education Occupation 0.0026379469

bn.graph.cpt  <- bn.fit(dag,survey,method="bayes")
bn.graph.cpt
```

```
##
##   Bayesian network parameters
##
##   Parameters of node Age (multinomial distribution)
##
## Conditional probability table:
##
##        Education
## Age          high       uni
##   adult 0.4423423 0.5404762
##   old   0.2504505 0.1047619
##   young 0.3072072 0.3547619
##
##   Parameters of node Residence (multinomial distribution)
##
## Conditional probability table:
##        big     small
## 0.7529412 0.2470588
##
##   Parameters of node Education (multinomial distribution)
##
## Conditional probability table:
##
##          Residence
## Education       big     small
##      high 0.6888021 0.8373016
##      uni  0.3111979 0.1626984
##
##   Parameters of node Occupation (multinomial distribution)
##
## Conditional probability table:
##
##           Education
## Occupation       high        uni
##       emp  0.97432432 0.91071429
##       self 0.02567568 0.08928571
##
##   Parameters of node Sex (multinomial distribution)
##
## Conditional probability table:
##
##    Education
## Sex      high       uni
##   F 0.3581081 0.5250000
##   M 0.6418919 0.4750000
##
##   Parameters of node Transport (multinomial distribution)
##
## Conditional probability table:
##
##          Residence
## Transport        big      small
##     car   0.58506944 0.54497354
##     other 0.19965278 0.09259259
```

Figure 2.1: Example visGraph display



```
##     train 0.21527778 0.36243386
```

Next we get things all ready for display via visNetwork and Shiny, but I can't evaluate the code here as it needs the Shiny machinery on a web server so I can't print to a page. However, the resulting network is shown in a screen print below. This code does work as-is within RStudio as it provides a viewer that shows the resulting graph.

```
edge.widths  <- max(arc.strengths$strength)/arc.strengths$strength
# Turn bnlearn "arcs" into visNetwork "nodes" by doing a lookup from names to IDs
visNetwork.edges <- data.frame(
        from=visNetwork.nodes$id[match(arcs(bn.graph)[,'from'],visNetwork.nodes$label)],
        to=visNetwork.nodes$id[match(arcs(bn.graph)[,'to'],visNetwork.nodes$label)] ,
        arrows="to",  # define the arrow style
        width=edge.widths)
visNetwork(visNetwork.nodes, visNetwork.edges) %>%
  visOptions(nodesIdSelection = list(enabled=TRUE,selected=1), highlightNearest = TRUE) %>%
  visHierarchicalLayout(sortMethod = "directed")
```

Here is the resulting graph, with arc thickness indicating the strength of the contributions of the arc to the NPT for the target node. If you look at the interactive version on Shiny, you can click on the nodes to display the CPTs and highlight the parents and children.

For the Shiny code, see GitHub.

2.2.3 The point of Bayesian Networks

When looking at just six variables, as in the example, if they are all independent then a table representing the probability of all possible states has a lot of entries, even when each can only take a small number of values. In this case it would be 143= (product of number of values for the 6 variables)-1). It would take a 6-dimensional table, which is beyond my powers of visualization. However, by first finding the dependencies, and representing it as a graph with lower-dimensional CPTs for each node, you cut things down significantly. In this case, from 143 free parameters to only 21.

This ability to factorise from a large multi-dimensional table to a collection of small ones is associated with the *Markovian property* of a Bayesian Network: given its parent variables, each variable is conditionally independent of other variables other than its descendants. I'll get more into the math in the next version of this. It took a while for all the symbols to suddenly gel in my little brain to make sense. I think I can write it down now in my own terms with a few pictures.

Chapter 3

Shiny

Shiny is an R package that lets you create a web application very quickly. RStudio lets you run this on the desktop with the click of a button and also provides a site where you can register and run a public application. There is a free version for low volume and a sliding scale of paid options for more “air time” and more space. And you just have to click a button in RStudio to deploy. I’ve never seen it so simple. The only thing that’s missing is a layout tool; you need to code that by hand, but it’s not very complex. However, you do have to remember to structure your application a little differently and debugging is a little harder because there is more hidden machinery between the parts of your application so you can’t always see exactly what is going on. The tuturial at http://shiny.rstudio.com/tutorial/ is excellent, but I haven’t yet done the more advanced stuff with reactive expressions and observers.

Conceptually, there are three parts to a web application, using the classic design pattern “Model, View, Controller”. The pattern says “keep these three things separate, in your code and your brain”. See, for example, this Wikipedia article. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Model is the code and data that represents the real-world things that you are representing. In this case, the bnlearn graph with its nodes, arcs and CPTs

View is the user interface with its panels, buttons and other widgets that the user sees and interacts with. Also includes PDFs, plots and other representations.

Controller is the code that mediates the interactions between the two, turning a click or some data entry into the right function call to update the model.

Generally, I add another piece

Data, which is the persistant data behind the model. For example, the raw data file “survey.txt” that I read in above, or a SQL database.

For single-use R scripts that you will not use after your data analysis, this may be overkill, but if you intend to maintain it over time, the initial investment in separating the concerns will probably pay off.

Appendices

Chapter 4

Tools

4.1 Why R?

I originally planned to use Python, rather than R, because I thought it a more general-purpose language and more suited to interactive programs

However:

- Some Python packages work only with 2.x, others with 3.x and I couldn't get the ones I wanted to work together
- I didn't know R did interactive. It does, very well, with Shiny and RStudio gives free/cheap hosting
- There are lots of books and web sites that use R for their examples and exercises. Many are cited here.
- R has three (slightly weird) object-systems and good support for functional programming. Looks pretty general purpose to me.

I'll look at Python again, later, in the hopes that some more packages have been updated, or I have time to find a better set, or I have time to do as they suggest: "port them yourself". In any case, there are many other tools in Python that I would like to try, but I need to do at most 42 things at once..

4.2 Graphics editor

The first diagram (1.1) was drawn using yEd. This exports in GraphML a standard that several graphics packages can read, including igraph (R and Python). This lets you do several formal diagram types for Information Technology, including Unified Modeling Language. Recommended. Free, from http://www.yworks.com/products/yed. I found it easy to import GraphML from yEd into R by using `igraph` functions, and then use that as the foundation for `bnlearn` analysis.

Until I found `visNetwork`, `igraph` was my favourite package. Lots of good graphic options, plus functions to transform from other representations, such as GraphML. From CRAN, or other sources for Python, C++.

4.3 Documentation

This document was written using LyX, an Open Source "What you see is what you mean" editor for LaTeX. Although originally designed (by Donald Knuth) for mathematical typesetting, I've pretty much managed to replace Word with LyX for my own writing of any kind. https://www.lyx.org/

Principal sources

I'm using these books as my main sources, supplemented by many places on the Internet that I will cite individually in the relevant places.

- "Réseaux bayésiens avec R"[3] A text on Bayesian Networks with R. I am finding this very useful as it has worked examples and exercises. There is an English version, but that was a lot more expensive.
- "Causality: Models, Reasoning and Inference" [4] THE book on this topic. Strongly recommended, though the math may be heavy going if you're not used to mathematical notation. However, I found that where I did get stuck, the next book got me out of trouble, together with working the R examples in the previous.
- "Risk Assessment and Decision Analysis with Bayesian Networks" [5]. Has really good introductions to statistics and probability before it dives into Bayesian Networks. It explains in great detail so is useful when I don't get something that Pearl is saying. Also gives a code to get free software which is useful for exploring this if you don't want to use a programming language. The software does all the work through an easy GUI. Recommended both by Judea Pearl (see above) and me.

For learning R as an n^{th} programming language, I found that the web version of Hadley Wickham's "Advanced R" [6] was mostly all I needed, plus of course helpful web sites, especially StackExchange.

Shiny has an excellent tutorial at http://shiny.rstudio.com/tutorial/ and points to a more in-depth one at zevross.com.

To do list

1. This document needs more context for other readers, which will include Future Eric all too soon.
2. Start to learn how this works with continuous variables
3. More advanced Shiny - see if I can produce a model behind all the textual descriptions of how the observers etc work. Possibly related to the "Observer" pattern from Gang of Four?
4. Look into OpenBugs
5. Biggest puzzle so far: how does this work for feedback loops? All the theory is for *acyclic* graphs, but there is causal feedback in most interesting complex systems. I think this comes in when I move past static into temporal networks.
6. Smaller puzzle, how do we get from mere correlations to causality. Since that is the title of the Pearl book, I'm hoping to find the answer before my brain gets too full.

Bibliography

[1] Marco Scutari. bnlearn: Bayesian network structure learning, parameter learning and inference. URL https://cran.r-project.org/web/packages/bnlearn/index.html.

[2] Almende B. V. (vis js library in htmlwidgets/lib, http://visjs.org, http://www.almende.com/home), and Benoit Thieurmel (R interface). visNetwork: Network visualization using 'vis.js' library. URL https://cran.r-project.org/web/packages/visNetwork/index.html.

[3] J. B. Denis and M. Scutari. *Réseaux Bayésiens avec R: Élaboration, Manipulation et Utilisation en Modélisation Appliquée.* URL http://discovery.ucl.ac.uk/1384681/.

[4] Judea Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press. ISBN 978-0-521-89560-6. URL http://www.cambridge.org/ca/academic/subjects/philosophy/philosophy-science/causality-models-reasoning-and-inference.

[5] Norman Fenton and Martin Neil. *Risk Assessment and Decision Analysis with Bayesian Networks.* CRC Press. ISBN 978-1-4398-0911-2.

[6] Hadley Wickham. Welcome · advanced r. URL http://adv-r.had.co.nz/.