The max-min-hill-climbing algorithm

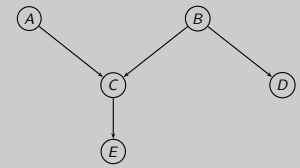
Michael Bauer
M.Sc. Comp. Science

July 14, 2014

- Hello, I wanna give you a warm welcome to my today's presentation. I will give you a short and nice overview about my project I worked on for the las 3 months.
- I had my focus on one of the state of the art algorithms in statistical computing: The max-min-hill-climbing algorithm.
- My task was to implement this in R or C or both. I was free to choose, which language I take. The only requirement was that this function could b called and run in R. At the end I decided to do both.
- Soon I learned, that there already was an implementation with all these requirements in a huge and really intelligent package called bnlearn, so I decided that the task should change to: Can I find an implementation whic is faster!?

Definition
A Bayesian Network is a directed acyclic graph (DAG) whose nodes are random variables and edges represent conditional dependencies. If two random variables are connected they are said to be dependent. If there is no connection they are said to be conditional independent.
For instance, we say: "A and B are conditional independent given C".

- Before I can give an answer to this question we have to go 2 steps back.
- We start where I started about 100 days ago: What does this algorithm actually do? And the answer to this question is, that it reconstructs Bayes'ian Networks from given data.
- How this is gonna work I'll tell you in a few minutes. First let me explain what a BN is.
- A Bayes'ian Network is a directed acyclic graph whose nodes are random variables and whose edges represent conditional dependencies.



- directed edges
- free of cycles
- random variable is represented as a node
- edges encode dependencies

- For instance, this network is a Bayes'ian Network. It is directed, which means all edges have only one arrow, it is acyclic since there are no cycles i it. The nodes represent random variables which can be connected. If there a connection between two nodes, then they are said to be dependent. Two nodes which are not connected are said to be conditional independent. In this case we say A and B are conditional independent given C.
- Of course, this introduction to Bayes'ian networks is simple and doesn't cover the whole complexity of those networks. But without going deeper into BNs you could already guess why those networks are so important.
- Because after constructing BN out of data, you see how this data connects See this example.

Predicting the effect of missense mutations on protein function: analysis with Bayesian networks

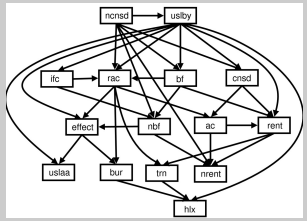


Figure: http://www.biomedcentral.com/1471-2105/7/405/figure/F2?highres=y (by Chris J Needham1, James R Bradford, Andrew J Bulpitt, Matthew A Care and David R Westhead)

- This graph for instance is a Bayes'ian network which I got from the paper "Predicting the effect of missense mutations on protein function: analysis with Bayesian networks" by Chris J Needham1*, James R Bradford2, Andrew J Bulpitt1, Matthew A Care2 and David R Westhead2. (http://www.biomedcentral.com/1471-2105/7/405)
- Of course, our time is too short to cover all the interesting things which come from this paper. But I'll leave you with the idea behind it:
- Predict the effect on the protein function. This research has a bioinformatical and a pharmaceutical matter. But to think, that this is only a biological "thing": Here are some examples where BN are used.

Figure: http://www-ekp.physik.uni-karlsruhe.de/Zupanc/WS1011/docs/Datenanalyse2010_3.pdf

- In medicine and sport they have a common use. AC Mailand football club tried to predict injuries of players.
- But also in economics those networks are use.
- At the end of the day this has an effect on all of us!!
- But how can we now reconstruct such graphs?

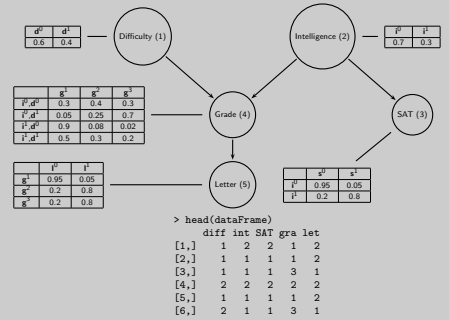
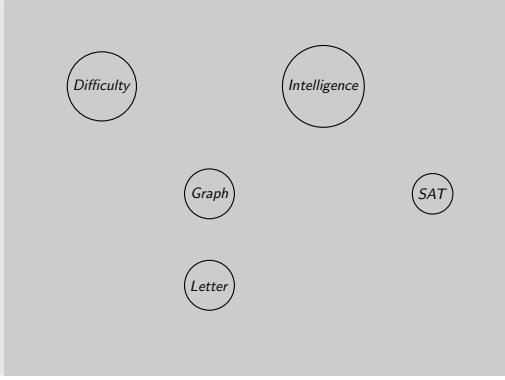
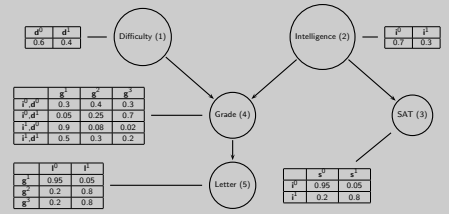


Figure: The data we observe from following the rules above.

- We start from observational data. From experiments or databases we have huge data. We assume that this data is complete.
- In my case I had data from an example out of the book of Daphne Koller.
- Explain my example.
- What my algorithm now does is as follows:

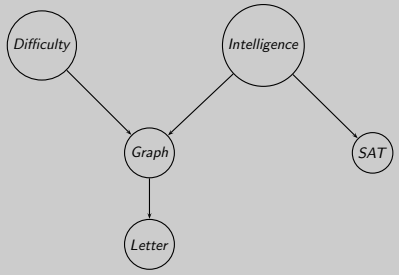


- With statistical methods it discovers if nodes are connect (dependent) or no (independent). It detects whether two nodes have strong dependencies or not.
- But if two nodes are connected by only looking at their data, the algorithm maybe finds out that they have no connection when looking at the combination of more data.
- If they have then this connection is saved in a List.
- Where you end up is, that you know all the connected nodes in a graph. Bu you don't know yet, in which direction the arrow goes.
- But once you have the structure of a graph you did half of the work.
- To show that my algorithm is valid and returns the correct results you can have a look at the output of the corresponding function:



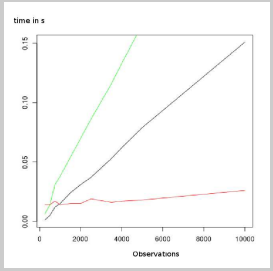
- With a score called BDeu (Bayesian Dirichlet equivalent uniform) score we discover our arrow directions.
- What this does is: It starts from an empty graph, so only nodes without edges. This empty graph gets a score.
- Then you add an edge - if the two nodes have a connection from MMPC - and give the whole graph a new score. If this score is better, then the edge stays, else it will be removed.
- You also reverse and delete.
- You do that as long as the graph's score does not change some time.
- At the end you have to check whether you have cycles in it. If so: remove an edge from the cycle.
- The validity of my code is shown on the next slide.

Bayesian Dirichlet equivalent uniform score



- As I said at the beginning, I tried two implementations to have a better comparison and a better chance to beat the bnlearn package.
- Only one word about this package: It was a Ph.D. project which does not exactly do the same as my code. It is intelligent and can decide, depending on the data, which algorithm and score fits best to the data.
- Trying to beat this was a bit of excitment and it was challenging.
- So, after programming most of the stuff in R, I discovered the following:

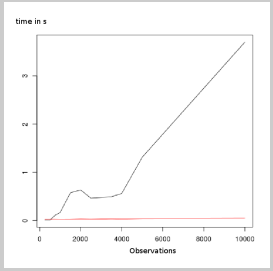
Computing the skeleton



nobs	R	C	bnlearn
250	0.006	0.001	0.014
500	0.014	0.005	0.014
750	0.031	0.012	0.017
1000	0.038	0.015	0.014
1500	0.054	0.024	0.015
2500	0.086	0.037	0.019
5000	0.166	0.079	0.018
10000	0.363	0.151	0.026

- Ok, as you can see: since a big part of the bnlearn package is implemented in C, my algorithm in R didn't have any chance to beat the package.
- Then I tried to bring all the code to C and optimized it a bit. Migrating cod was not that easy as somebody could think, because R has a big amount o functionality and nice little widgets which help to do things in an easy way.
- After some hour the implementation in C++ was completed and the new benchmark looked as follows:

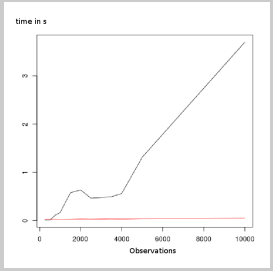
Computing the edges



nobs	C	bnlearn
250	0.011	0.023
500	0.007	0.024
750	0.107	0.024
1000	0.166	0.022
1500	0.575	0.023
2500	0.493	0.036
5000	1.313	0.041
10000	3.701	0.048

- By taking a closer look to my code and thinking where the time gets lost, have to say, that all those statistical computings I do are pretty fast for sma data (sometimes faster than bnlearn), but as the data grows - which is of course the case in genomic experiments - the runtime gets worse.
- After benchmarking several functions I discovered that there is not only on fucntion which takes the whole time by dealing with big data.
- At the end it's the sum of all functions which make it slower than bnlearn. But as I said this package is highliy optimized.
- So, to conclude this I think more experience in programming and more tim would be necessary to get a better chance to beat it.

Computing the edges



nobs	C	bnlearn
250	0.011	0.023
500	0.007	0.024
750	0.107	0.024
1000	0.166	0.022
1500	0.575	0.023
2500	0.493	0.036
5000	1.313	0.041
10000	3.701	0.048

- At the end I want to say thank you to Giusi who always helped me, when help was needed.
- I also wanted to thank Tomi Silander. A scientist who replied to my email after 15 minutes, answered all my questions, send me his code and brought his knowledge to me via email in a very simple and understandable way.
- And to you: Thank you for your attention!!!