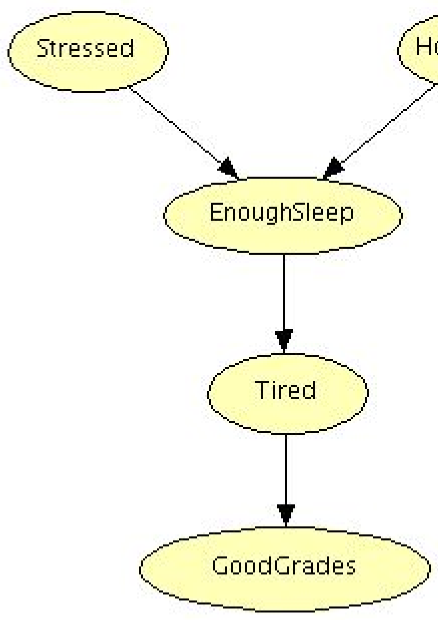
Gustavo Gutiérrez
Franco Valencia

Lab 4 Report

Implementing Bayesian Networks

First one of the team members did not had much experience with Python. So it took a time to get comfortable with it. Then we faced some challenges parsing the input, since we had to take certain lines, and parse such lines in a way that we could manipulate them easier.

Used Bayes Network



Implementation vs Hugin Lite

Query	Own implementation	Hugin Lite	Match?
P(+GG)		63.08%	
P(+GG|-ES,-H)		66.58%	
P(+S|-GG)		40.04%	
P(-ES|+H,+T)		89.36%	

Gustavo Gutiérrez
Franco Valencia

What are the differences between what they generate?

The differences are just of information presentation. In one hand, the Hugin program presents a list of all the nodes and an easy way to see their interaction if we “give” some nodes a given state, by clicking them; the resulting probabilities show up immediately.

In the other hand, our implementation lacks of a GUI, and the program runs just once, as it reads all the information it needs to run: nodes, probabilities and queries. Then it just makes a BN that is not visible, and calculates each probability for its given query.

Do they use the same algorithms?

No, the Hugin Lite software uses what they call *Hugin Decision Engine*, while our implementation uses the *Enumeration Algorithm*.

Taken from Hugin’s website, the HDE is defined as an engine that performs reasoning on a knowledge base represented as a Bayesian Network or LIMID (limited-memory influence diagrams). The HDE performs all data processing and storage maintenance associated with the reasoning process. One of the most important components of the HDE is its compiler, which transforms the BN into structures called *junction trees*, which make the analysis very efficient. The Hugin team says that for small and medium sized networks, inference takes just a fraction of a second, and for large networks (they don’t specify how many nodes make a “large network”) the analysis takes just a few seconds.

On the other hand, our implementation uses the Enumeration Algorithm, which makes a new inference with each query that the program receives. In case the algorithm takes a recursive depth-first approach and according to a presentation made by the Simon Fraser University (https://www.cs.sfu.ca/~mori/courses/cmpt310/slides/chapter14b.pdf), the complexity is:

- O(n) on space
- O(d^n) on time

This algorithm is a slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation and storing it in a data structure.

What are their common bases?

The common base in both approaches to solve queries given a set of different probabilities and their dependencies is the representation of a Bayes Network. This representation decreases drastically the number of entries in a probability table that are needed to make a valid inference.

In terms of the Bayesian Network previously proposed, if we need to capture the joint probability, the number of needed parameters are:

- 31 without Bayes Network representation
- 10 with Bayes Network representation

The number of needed parameters without a BN representation grows exponentially, in the order of 2^n-1. This is unsustainable when a the network has a big amount of nodes.

Which tool would you use for what cases in real life applications?

It really depends on the application and the given time constraint. If we had a scenario on which it is important to infer some probabilities fast, we would use a combination of both implementations, using the Hugin Decision Engine but in a program without a GUI, in which it would be possible to read the BN representation as we read it in our implementation (text files) and output the probability of the queries.

We don’t say Hugin is bad, actually their implementation is really good and efficient. The problem is the slowness brought by using the GUI. On the other hand, if it is for educational purposes, we would definitely use the Hugin Lite program, which helps users to picture the Bayes Network they are working with.