Evaristo Koyama
ek4ks

Bayesian Belief Networks

Bayesian Belief Networks are incredibly useful tools. They make data easy to understand and can make accurate predictions. They are used in a wide variety of disciplines, ranging from making medical diagnosis to analyzing advertising data. For the class project, I developed a simple tool in python that can construct BBN based on the algorithm of Cheng, Bell, and Liu and maximum likelihood. It was tested using the Adult dataset. It is popular dataset for constructing models. It is built form US census data and has a total of 14 variables, both discrete and continuous.

A Brief Overwiew

A Bayesian belief network is a directed acylic graph representing cause-and-effect relationship between variables. A parent node is said to "cause" its child node. In mathematical terms, each node is the probability of a value in the domain of child node given a set of values in the domain of the parent nodes. This powerful notation allows us to calculate the joint probability distribution without using an exponential amount of space. If we were to store the distribution of a set of k discrete variables, then we need to store at least 2^k probabilities assuming all of them are binary. With a Bayesian network, each node only has to store information of the values it and its parents takes. To calculate the joint distribution, we can take the product of all the nodes in the network starting from the root. We assume independence for the variables that are not directly connected. With a properly constructed network, we can mitigate some of the error caused by making this assumptions.

Project Overwiew

The tool is divided into 4 steps. Data input, data parsing, network building, and parameter estimation.

Evaristo Koyama
ek4ks

The first step is data input. We want the set of variables names, the values a variable can take on, and the actual data. This step is simple enough. There is also an option to save the data at a given step. Some steps make take awhile based on the parameters and the dataset

The second step is data parsing. In this step we can take a preliminary glance at some of the data. This tool allows you to look at the frequency of each value of a categorical variables. With this in mind, you can decide whether you want to limit the values of a categorical variables. There are usually two reasons for this. The first is that the value does not appear frequently. Extreme values can affect the performance of the network. Second is if it can take on too many variables. The size of the conditional probability table of each node can increase exponentially based on the number of parents and the number of variables each parent can take. In the Adult dataset example, we manually transform the education variable and the native-country. We take all value that take on ["1st-4th", "10th", "9th", "7th-8th", "12th", "11th", "5th-6th", "Preschool"] and cover it to No-HS. You can make some educated guesses when deciding to transform the data. Native-country was changed to a binary value of United-States and Foreign. The dataset did not contain many values outside of the United-States. The tool also has an option of discretizing continuous variables. We can make bins based on percentile. The tool calculates the appropriate number of bins based on how many you want and the distribution of the data. In the example dataset, capital-gains and capital loss is frequently 0. Even if we want to use more than 2 bins, the tool still only produces 2 bins. It's still possible to make more bins, if you decide to increase the percentile division, but too many bins means a large conditional probability table. The tool can then transform the dataset based on your categorical variable change and discretization.

The third step is network building. We use the algorithm based on Cheng, Bell, and Liu. We first calculate the mutual information between pairs of variables in the dataset. The value of mutual information is lowest when the pair of variables is independent. The idea is that we want to put related

information close together. We sort the pair of variables based on mutual information from highest to lowest. We eliminate any pair of variables based on threshold that can be set. Setting the threshold lower will increase the amount of time it takes and produce a more denser network. We randomize the order of the variables. An option can be used to force an order or force connections with a variable. This option can be useful if you want to focus on particular variables. We then follow the algorithm. We first add edges to the network if the pair of variables are independent given no variables. Conditional Independence is tested using on d-separation. We first check if there is an undirected path available between the pair of variables. If there is no path, then there are independent. Otherwise, we move on. We first make a graph of the ancestors of all the variables used. This includes the given variables. We then add edges between pairs of variables that share a child node. When we check for shared child nodes, we temporarily add and edge connecting the pair of variables we are testing. We then remove all the edges surrounding the variables that are given. We then check if a path exist between the two variables. If so, we declare them independent. Otherwise, it's not. After testing the conditional independence of the pairs of variables. We get a preliminary network. In the next step, we add more edges based on conditional independence based on the current structure. In this step, we find a blockset, the set of nodes that can separate a pair of variables. We use breadth first search to find the minimum length paths. Multiple paths can be chosen if the paths don't overlap. From the path, we pick the appropriate node to use for blocking. This is dependent on the direction. If the path is a direct path between the variables, then we take the node closest to the end assuming the path is not already blocked. If the path has a change in direction, then we take the node where the path changed direction. We then use the set of nodes to block the path between the pair of variables, meaning we test conditional independence given the blockset of nodes. The final step is pruning. We test each edge using a similar strategy as the previous step. We remove the edge and check to see if the pair of

variables is still dependent based on the current structure of the network. If it is, then we add the edge back.

The last step is estimating the network parameters. We use maximum likelihood estimation to calculate the values in the conditional probability tables. For each node, we go through the data. If the node has no parents, then we just calculate the amount of times a value showed up and divide it over the size of the dataset. If the node has parents, then we count the amount of times a value showed up given the set of parent values. We then divide it by the number of times the set of parent values showed up.

References

Dataset: https://archive.ics.uci.edu/ml/datasets/adult

Condtional Independence Test Algorithm: http://web.mit.edu/jmn/www/6.034/d-separation.pdf

Causal Belief Network Overview:
https://pdfs.semanticscholar.org/ed54/3eedde24edd50e39f5567f036b632499240a.pdf

Building Bayesian Network Overview:
https://pdfs.semanticscholar.org/6bce/1db9b6d7d49b8c4ce286f025ace33bb3d31e.pdf

Bayesian Parameter Estimation: http://www.cse.ust.hk/bnbook/pdf/l06.h.pdf

Bayesian Inference Tutorial: https://www.norsys.com/tutorials/netica/secA/tut_A1.htm

Bayesian Network Construction: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.587.3852&rep=rep1&type=pdf