Project in Bayesian Networks

Uzi Smadja - 313673782 - uzi.smadja@gmail.com

Bernie Howard - 203039417- bernie.howard@gmail.com

Problem Description

This project will consider forest fire data from the Montesinho natural park from the Trás-os-Montes northeast region of Portugal (Figure 2). The park contains a high flora and fauna (plants & animals) diversity. In the simplest case, a Bayesian network is specified by an expert and is then used to perform inference. In other applications the task of defining the network is too complex for humans.

In this case the network structure and the parameters of the local distributions will be learned from data itself. Following the learning of the structure and distributions, we intend to evaluate and compare between two of the inference algorithms learned in class. We will explore the difference between exact inference and approximate inference.

Fire Weather Index:

The forest Fire Weather Index (FWI) is the Canadian system for rating fire danger and it includes six components (Figure 1): Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC), Drought Code (DC), Initial Spread Index (ISI), Buildup Index (BUI), FWI

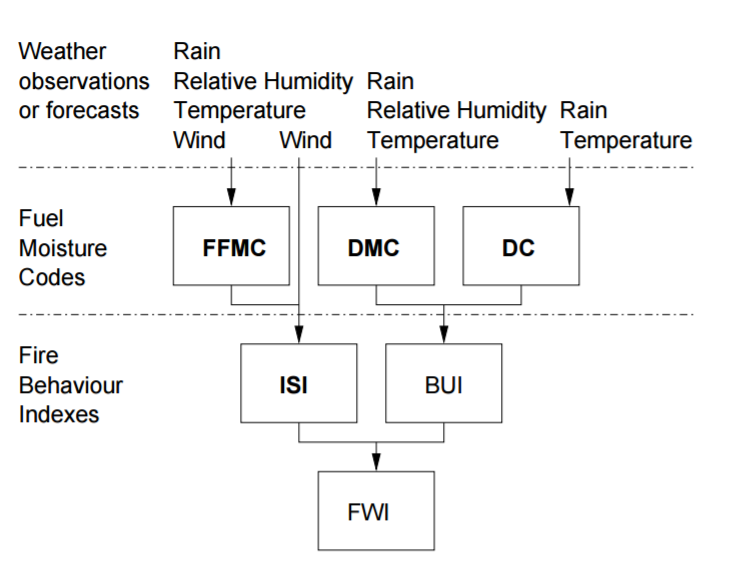
****

Figure 1 Fire Weather Index structure

The first three are related to fuel codes: FFMC denotes the moisture content surface litter and influences ignition and fire spread, while the DMC and DC represent the moisture content of shallow and deep organic layers, which affect fire intensity. The ISI is a score that correlates with fire velocity spread, while BUI represents the amount of available fuel. The FWI index is an indicator of fire intensity and it combines the two previous components. Although different scales are used for each of the FWI elements, high values suggest more severe burning conditions. Also, the fuel moisture codes require a memory (time lag) of past weather conditions: 16 hours for FFMC, 12 days for DMC and 52 days for DC.

Dataset:

The data used in the experiment was collected from January 2000 to December 2003 and it was built using two sources. The first database was collected by the inspector that was responsible for the Montesinho fire occurrences. At a daily basis, every time a forest fire occurred, several features were registered, such as the time, date, spatial location within a 9×9 grid (x and y axis of Figure 2), the type of vegetation involved, the six components of the FWI (Figure 1) system and the total burned area. The second database was collected by the Bragança Polytechnic Institute, containing several weather observations (e.g. wind speed) that were recorded with a 30 minute period by a meteorological station located in the center of the Montesinho park. The two databases were stored in tens of individual spreadsheets, under distinct formats, and a substantial manual effort was performed to integrate them into a single dataset with a total of 517 entries.

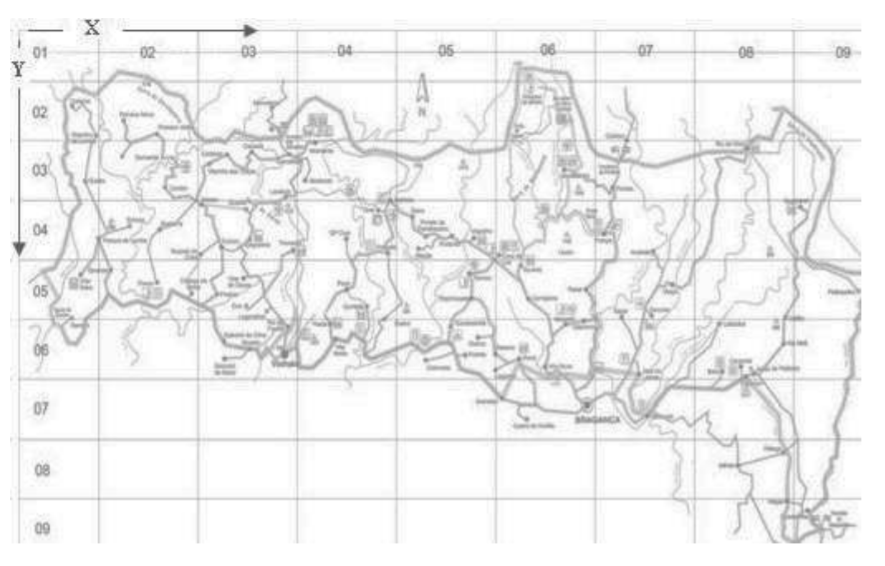


Figure 2 The map of the Montesinho natural park

Bayesian Network Structure Learning

Databases

As described, for start we had a 517-entry full dataset, each entry containing the following thirteen fields:

* Longitude and latitude coordinates as shown in figure 2
* Gregorian month and day of the week of the forest fire occurred
* Weather conditions such as: temperature, humidity, wind and occurrence of rain during the day
* FFMC, DMC, DC, ISI as defined earlier
* Area of the forest fire

In addition, we created another small database, each row within it contains an “experts” information about a certain fire event that could happen “by logic” - For example, a large fire in the middle of the forest, during a hot dry summer day with no rain.

We quantized values of the database [UZI – EXPAND]

Model Learning

To simulate a Bayesian graph, we created a graph class containing the node classes as mentioned above, each node contains a set of parents. As seen on the graph in fig. 1, some father-child node dependencies are already known, but not all of them (i.e. X and Y coordinates aren’t presented in fig. 1).

We decided to create an initial graph as a base for our calculations and assumptions (fig. 3). We implemented two simple graph-changing algorithms, both changing the graph by adding edges, removing edges and switching directions of directed edges.

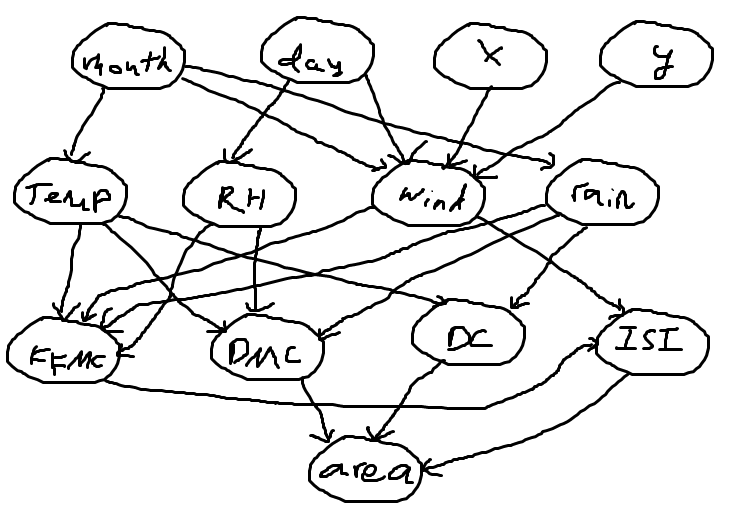
The first method is manual, meaning a file containing add/remove/switch commands is given as an argument to the program. The second method is random manipulating edges inside the graph. Both methods can of course damage the DAG characteristic of the graph, so a DFS is being executed before an add or switch command.

Fig. 3 Initial graph

Our goal is achieving the maximal likelihood estimation (MLE) of the model given the data above, meaning . As learned, , while . A grave assumption (prior) lies here (which we will attend to in the result section below), which is that the model structures are distributed uniformly, so was the goal searched.

As shown in lecture no. 10,  
When:

is the number of cases when and

is the number of states of

is the number of instances of the parents of

, and

We made a small change, and instead we did a log-likelihood estimation, denoted

Results

First, we’d Like to attend the second method we used, i.e. the random manipulation of the model. The auto generator function gets two arguments, I and R (iterations and rounds).

For each round , , we do a series of random changes in the model: 2I add operations, I removals and I swaps of edges. After each operation, the maximal-scored-model is saved and passed for further changes.

We tested the algorithm for different increasing I and R values, and got the following graph representing score as a function of number of operations (. Remembering that due to proximity to zero, we cannot represent the real score, each score is in fact .

The score in increses with an increasing number of iterations, but we notice that the graph manufatured isn’t always logic and strange dependencies occur, Such as the gregorian month of the fire is dependent by the humidy (and not vice verca). Such strnage dependencies can be explained due to the fact of uniformity of the graph probability assumed earlier. In fact, the probablity of a graph with a dependency like the one presented above is close to zero.

Considering these problems, we approached the problem with a more careful approach and with consideration to the initial graph. We manually created subtle changes in the graph and checked the score.

This approch is far more accurate in our opinion, and we did see that swapping edges in a non logical way did harm the model score, i.e. the probability for such model is worse.