Machine Learning
Artur d'Avila Garcez
5th Tutorial – Bayesian networks

1. Bayesian Networks

Background

A Bayesian network, Bayes network, belief network, Bayesian model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Formally, Bayesian networks are DAGs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes that are not connected represent variables that are conditionally independent of each other. Each node is associated with a probability function that takes, as input, a particular set of values for the node's parent variables, and gives (as output) the probability (or probability distribution, if applicable) of the variable represented by the node. For example, if m parent nodes represent m Boolean variables then the probability function could be represented by a table of 2^m entries, one entry for each of the 2^m possible combinations of its parents being true or false. Similar ideas may be applied to undirected, and possibly cyclic, graphs; these are called Markov networks. Finally, Bayesian networks that model sequences of variables (*e.g.* speech signals or protein sequences) are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

There are three main inference tasks for Bayesian networks:

Inferring unobserved variables

Because a Bayesian network is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them. For example, the network can be used to find out updated knowledge of the state of a subset of variables when other variables (the *evidence* variables) are observed. This process of computing the *posterior* distribution of variables given evidence is called probabilistic inference. A Bayesian network can thus be considered a mechanism for automatically applying Bayes' theorem to complex problems.

Parameter learning

In order to fully specify a Bayesian network and thus fully represent the joint probability distribution, it is necessary to specify for each node *X* the probability distribution for *X* conditional upon *X*'s parents. The distribution of *X* conditional upon its parents may have any form, but it is common to work with discrete or Gaussian distributions. Sometimes only constraints on a distribution are known; one can then use the principle of maximum entropy to determine a single distribution, the one with the greatest entropy given the

constraints. (Analogously, in the specific context of a dynamic Bayesian network, one commonly specifies the conditional distribution for the hidden state's temporal evolution to maximize the entropy rate of the implied stochastic process).

Often these conditional distributions include parameters which are unknown and must be estimated from data, sometimes using the maximum likelihood approach. Direct maximization of the likelihood (or of the posterior probability) is often complex when there are unobserved variables. A classical approach to this problem is the expectation-maximization algorithm which alternates computing expected values of the unobserved variables conditional on observed data, with maximizing the complete likelihood (or posterior) assuming that previously computed expected values are correct. Under certain regularity conditions this process converges to maximum likelihood (or maximum posterior) values.

Structure learning

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 must be learned from data.

Automatically learning the graph structure of a Bayesian network is a challenge pursued within Machine Learning. The basic idea goes back to a recovery algorithm developed by Rebane and Pearl (1987) and rests on the distinction between the three possible types of adjacent triplets allowed in a directed acyclic graph (DAG):

1. $X \rightarrow Y \rightarrow Z$
2. $X \leftarrow Y \rightarrow Z$
3. $X \rightarrow Y \leftarrow Z$

Type 1 and type 2 represent the same dependencies (X and Z are independent given Y) and are, therefore, indistinguishable. Type 3, however, can be uniquely identified, since X and Z are marginally independent and all other pairs are dependent. Thus, while the *skeletons* (the graphs stripped of arrows) of these three triplets are identical, the directionality of the arrows is partially identifiable. The same distinction applies when X and Z have common parents, except that one must first condition on those parents. Algorithms have been developed to systematically determine the skeleton of the underlying graph and, then, orient all arrows whose directionality is dictated by the conditional independencies observed.

A method of structure learning uses optimization based search. It requires a scoring function and a search strategy. A common scoring function is the posterior probability of the graph structure given the training data. The time requirement of an exhaustive search returning a structure that maximizes the score is super-exponential on the number of variables. A local search strategy makes incremental changes aimed at improving the score of the structure. Friedman et al. (1997) seek to find a structure that maximizes *mutual information* (a measure of how similar P(X,Y) is to P(X)P(Y)) between variables. They do this by restricting the parent candidate set to *k* nodes and exhaustively searching therein.

Setup

We will run a simple example of structure construction of a Bayesian network. We will use an algorithm called K2, which is based on hill-climbing over possible optimal structures. An artificial, binary dataset with 10 variables will be used for constructing the DAG.

1. Unzip “bnet.zip” into any folder of your choice;
2. Open Matlab and inside it, locate the folder where the contents of “bnet.zip” has been unzipped;
3. Double click on “bnet.m” inside Matlab;
4. Left-click the editor and press F5 to run the code

You should see a graph representing the optimal directed connections between the 10 variables, according to K2. A brief tutorial on K2 can be found at:

citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.190.7306&rep=rep1&type=pdf

Exercises

Change the order of the input and the maximum number of edges and inspect the changes in the graph.

Change the data set to include fewer data points; see if you can infer what K2 does in this way.

Open k2.m and check the code. Does it match your intuition from the previous exercise?