# Autonomous Agents

Bayesian Classification

Παπαγεωργίου Βασίλειος (AM: 2016030080)

*February 23, 2020*

## Contents

## 1 Introduction

In this project we study the case of a specific type of classifiers, called **Bayesian Classifiers**, a specific type of classifiers which make use of the Bayes theorem to assign labels to data, aiming to maximize the posterior probability. The aforementioned method was tested in the widely known and used **alarm network** as well as in a **medical network** which models the dependencies between causes and symptoms for a variety of breathing diseases. The outline of the report is as follows: In [2] we describe the fundamentals of bayesian networks. In [3] we set the basic framework of statistical learning and the general philosophy behind bayesian classifiers, in [3.1] we describe the process of utilizing bayesian networks for statistical learning using two specific black approaches and lastly in [4] we describe the proposed implementations and the simulation results.

## 2 Bayesian Networks

Bayesian Networks (BN) is a specific graphical model that is used to manipulate joint distributions of random variables, managing to efficiently model their dependencies, in a systematic way. More specifically, it consists of a directed and acyclic graph (DAG) whose nodes are random variables; each random variable $X_i$ has a conditional distribution $\Pr(X_i|\text{parents}(X_i))$ which is **only dependent on its parents in the graph** and is stored in a table called **Conditional Probability Table** (CPT). As a result, we can write:

$$\Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} \Pr(x_i|\text{parents}(X_i))$$

The most basic task for any probabilistic inference system - and as a result for BN's- is to calculate the posterior probability distribution for a set of query variables, given the observations of a set of evidence variables. This is called **exact inference** and there are various algorithms who calculate the answers in such queries. What we implement in this project is an algorithm called **variable enumeration**. This algorithm is based on the simple observation that any query $\Pr(X|\mathbf{e})$ can be answered by:

$$\Pr(X|\mathbf{e}) = a\Pr(X, \mathbf{e}) = a\sum_{\mathbf{y}} \Pr(X, \mathbf{e}, \mathbf{y})$$

where $a$ is a normalization constant, $\mathbf{e}$ the set of evidence variables and $\mathbf{y}$ the set of hidden variables.

For the purpose of generating artificial datasets, we also utilize an algorithm that approximately answers queries such the one that was described above, which belongs to the set of randomized **Monte Carlo** sampling algorithms, and is called **prior sampling**. What this algorithm suggests is that each variable is sampled according to the conditional distribution given the values of its already sampled parents.

## 3 Bayesian Classifiers

A statistical **classifier** is a function $f : \Omega_{\mathbf{X}} \to \Omega_C$ that maps the values of attributes $\mathbf{X} \in \mathbb{R}^n$ to a unique class label $c \in \Omega_C = \{c_1, \ldots, c_m\}$. In order to induce such a function, a dataset $\mathcal{D}$

with $N$ labeled tuples $< \mathbf{X}, C >$ is required, to which a variety of algorithms can be applied aiming to learn the function $f$ that best describes the data in $\mathcal{D}$ and generalizes well in the case of unseen data. The goal is to find a function $f$ that when given an observation $\mathbf{x}$, it assigns $c^*$ to it, where:

$$c^* = \underset{j}{\operatorname{argmax}}\{\Pr(c_j|\mathbf{x})\} \tag{1}$$

Bayesian classifiers are a specific case of statistical classifiers, as were described above. The term "Bayesian" stems from the fact that we use **Bayes' theorem** in (1) getting:

$$c^* = \underset{j}{\operatorname{argmax}}\{\Pr(c_j)\Pr(\mathbf{x}|c_j)\} \tag{2}$$

Usually, the space of $\mathbf{x}$ is of a fairly high dimension, which leads to the fact that the calculation of the term $\Pr(\mathbf{x}|c_j)$ in (2) is challenging. Hence, it is necessary to make some independence assumptions in order to lower the complexity of the system, which are described later in the report.

## 3.1 Learning Bayesian Networks

The Bayesian Networks' learning, consists of two phases, one to learn the structure of the DAG of attributes and one to estimate the parameters of the conditional distributions (the values of CPT's) using a specific approach (in our case **maximum likelihood**). More formally, given a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ and a training dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ our goal is to induce a BN that best matches $\mathcal{D}$.

### 3.1.1 Naive Bayes

Naive Bayes classifier makes the "naive" assumption that all the attributes are conditionally independent, given the value of the label. As a result, it holds that:

$$\Pr(\mathbf{x}|c) = \prod_i \Pr(X_i = x_i|c)$$

Hence, we can rewrite (2) as:

$$c^* = \underset{j}{\operatorname{argmax}}\{\Pr(c_j)\prod_i \Pr(X_i = x_i|c_j)\}$$

If we see this classifier as a BN, its topology is similar to the one represented in Figure 1, where the label is the only parental vertex of each variable, implying that all variables are conditionally independent given the label.

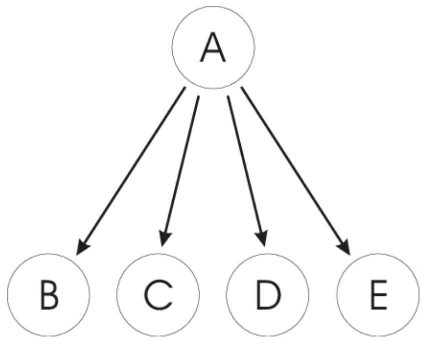

Figure 1: Naive Bayes classifier BN structure example.

### 3.1.2 Tree Augmented Naive Bayes

Tree Augmented Naive (TAN) Bayes classifiers loosen the tight restriction of conditional independence of all the attributes of the problem, by allowing to each variable to have at most one other parent other than the label in the network. The network of such a classifier can be seen in Figure 2.

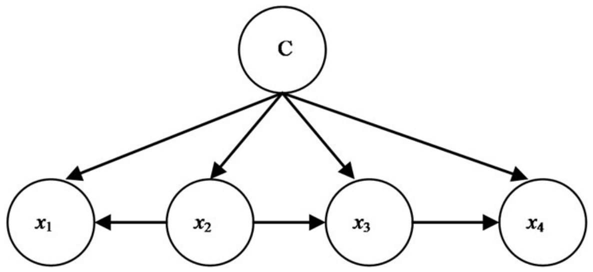

Figure 2: Tree Augmented Naive Bayes classifier BN structure example.

We can see that in contrary to the case of Naive Bayes classifier, the structure of the network is not know. As a result, we have to utilize an algorithm that is used for the task of learning bayesian networks' structures. The algorithm that we use is the **Chow Liu** algorithm, which is able to induce BN's topology when the number of parents of each node is restricted, while its goal being to find a **tree that maximizes the likelihood of the training data**.

First of all, we have to highlight that the **mutual information** of two random variables is given by:

$$I(X,Y) = \sum_{x,y} p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x)P_Y(y)}$$

What Chow Liu suggests is to firstly calculate the weight $I(X_i, X_j)$, $\forall i \neq j$, find a **maximum** weight spanning tree from the initially fully connected graph and make the graph directed starting from a random vertex using a traversal like BFS or DFS.

TAN classifiers make use of this algorithm to induce the dependency relationships between the different attributes but the label. More specifically, Chow Liu is initially used to all the attributes except for the label, using the conditional mutual information of the variables given the label as the weight. After that, the label is added as a parental vertex to each variable of the graph and the CPT's are calculated from the training data (in our case using **maximum likelihood** estimation).

## 4 Implementation

### 4.1 Code

The aforementioned techniques and algorithms were implemented using raw Python. The code structure is as follows:

- **main.py**: contains the driver code that is used to perform the experiments
- **graph_utils.py**: contains the implementation of various graph operations (topological sort, prim's algorithm)
- **bn_utils.py**: contains the implementation of a class named **BayesianNetwork** through which the representation as well as the **varibale enumeration** algorithm are implemented.
- **bn_classifier.py**: contains the implementation of the class **TANClassifier** through which we implement the TAN as well as the Naive Bayes classifiers.
- **bn_test.py**: contains code that calculates the classification error of a learnt network.

### 4.2 Experiments

The use cases that we use to test the implementation of the algorithms that were described above are the widely know **alarm** bayesian network that we also saw during the lectures as well as a **medical** bayesian network used to classify patients based on their symptoms. The topologies of the graphs of the two networks are in Figures 3 and 4 respectively.

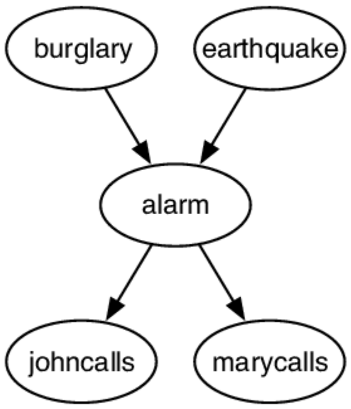

Figure 3: Alarm network.

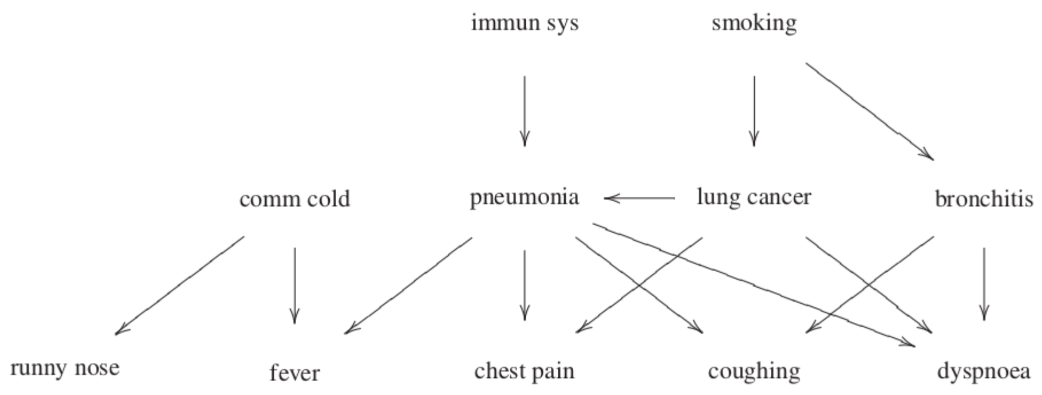

Figure 4: Medical network.

From these two networks, we make use of the prior sampling algorithm to generate two artificial datasets which are afterwards used to build a TAN and a Naive Bayes classifier. We then calculate the mean error percentage of misclassified samples in both cases, having a different label each time. The results are in Figures 5 and 6 alongside with the misclassification percentage that is yield by the initial networks.

We can see that in the case of the alarm network, both TAN and NB classifiers achieve to induce networks that model the random variables' dependencies well for every possible label case. As a result, the misclassification error is only slightly higher compared to using the actual BN, only in some cases.

Same conclusions can also be drawn from the case of the medical network. More specifically, both TAN and Naive Bayes reasoning is fairly close to the actual BN, with the Naive Bayes having difficulties to model random variables' relationships - and as a result has a higher error percentage- in the case where the label is lung_cancer.

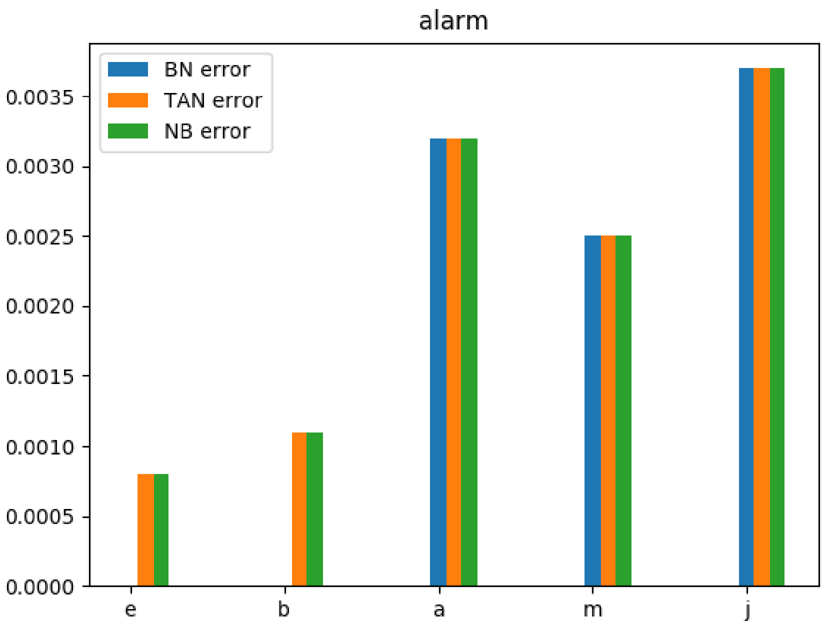

Figure 5: Alarm network error comparison.

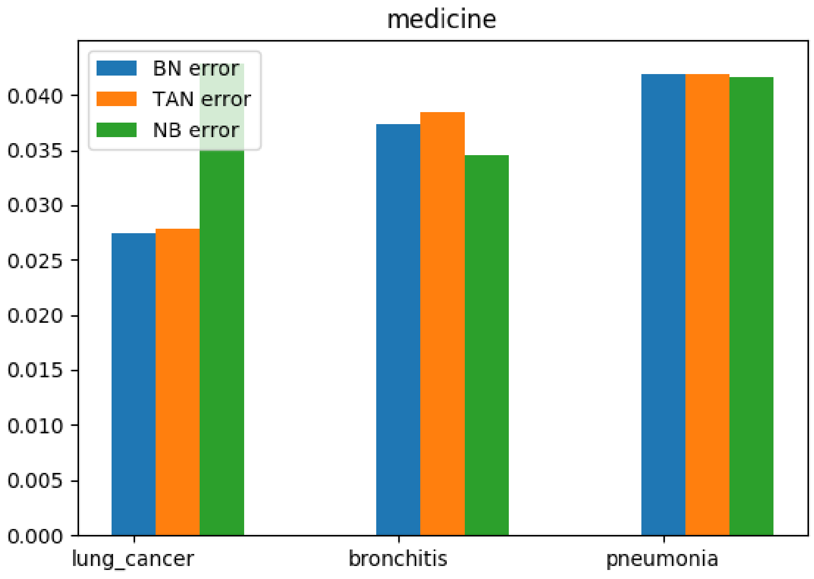

Figure 6: Medical network error comparison.