# Bayesian Classification

## Autonomous Agents

Vasilis Papageorgiou

February 23, 2020

Technical University of Crete

# Bayesian Networks

- Bayesian networks are graphical models that model **joint distributions** of random variables
- They consist of a directed and acyclic graph (**DAG**)
  - **vertices**: random variables
  - **edges**: random variable dependencies
- The **conditional probability distribution** of each random variable is only **dependent on the distribution of its parental vertices**:

$$\Pr(X_i | \cap_{j \neq i} X_j) = \Pr(X_i | \text{parents}(X_i))$$

- These distributions are stored in tables called **Conditional Probability Tables** (CPTs)

# Bayesian Networks

- As a result, for the joint distribution of all the random variables of the network it holds:

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

- Below we can see an example of a Bayesian Network:

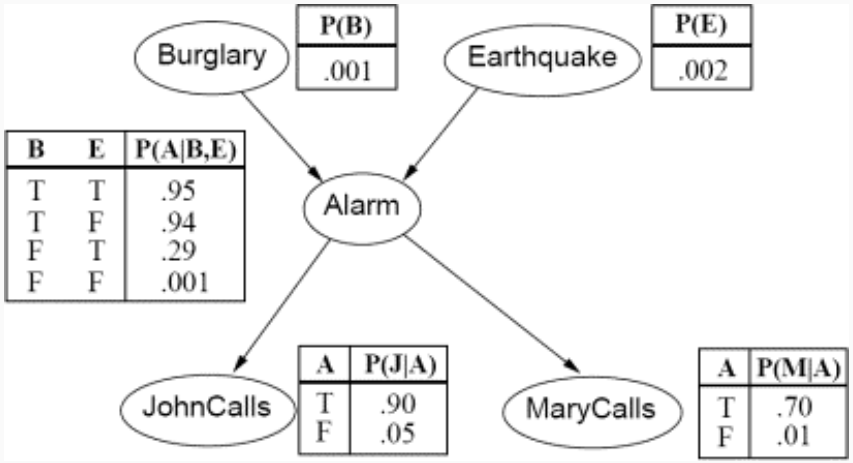



**Figure 1:** Alarm bayesian network.

# Exact Inference

- **Exact Inference** is the calculation of the posterior probability distribution for a set of **query** variables, given the observations of a set of **evidence** variables
- The algorithm that has been implemented is the **variable enumeration** algorithm:

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

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

# Bayesian Classifiers

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

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

- Using **Bayes' theorem**, we can rewrite the equation above as:

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

  which is the basis of every Bayesian Classifier

# Learning Bayesian Networks

- Given a dataset $\mathcal{D}$, learning a Bayesian Network consists of two phases:
  1. Learning the **structure** of the DAG
  2. Estimating the values of the **CPTs** (in our case using **maximum likelihood** estimation)

  the combination of which aims to induce a Bayesian Network that best describes $\mathcal{D}$.
- The challenge arises when the space of the attributes $\mathbf{X}$ is of a high dimension. In this case, the estimation of $\Pr(\mathbf{X}|c)$ is a hard task.
- The solution is given making some arbitrary random variable **dependency assumptions** that lower the complexity of the problem.

# Naive Bayes Classifiers

- **Naive Bayes** Classifiers make the tightest independence assumption:
  - All the random variables are **conditionally independent** given the value of the label
- Hence:

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

and as a result each label is assigned by:

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

# Naive Bayes Classifiers

- The topology of the DAG of such a classifier suggests that all the attributes of the problem have **only one parental vertex**, the one of the label random variable.
- Such an example is given below:

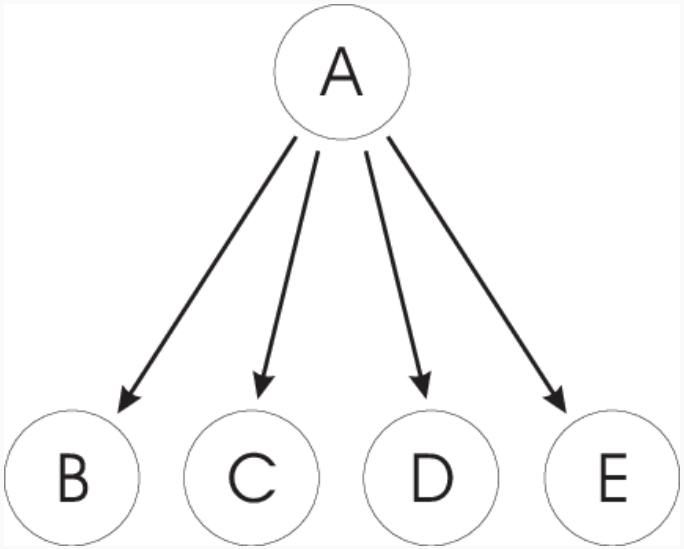



**Figure 2:** Naive Bayes Classifier DAG example.

# Tree Augmented Naive (TAN) Bayes Classifiers

- **Tree Augmented Naive** (TAN) Bayes Classifiers loosen the conditional independence that is suggested by the conventional Naive Bayes Classifiers
- They let each random variable to have **at most one parental** vertex, besides the vertex that corresponds to the label.
- An example of such a network is given below:

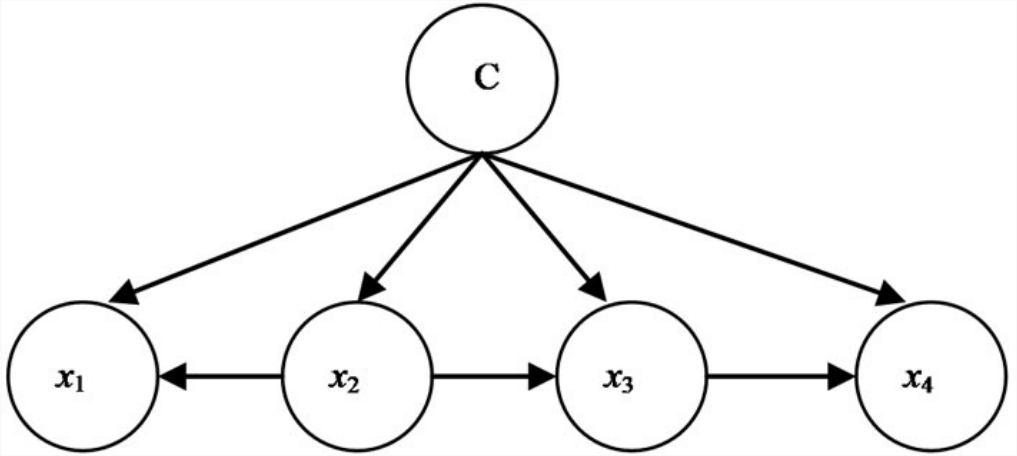


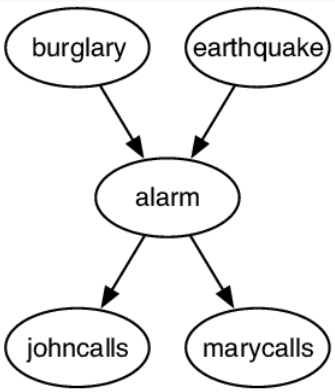

**Figure 3:** Tree Augmented Naive Bayes Classifier DAG example.

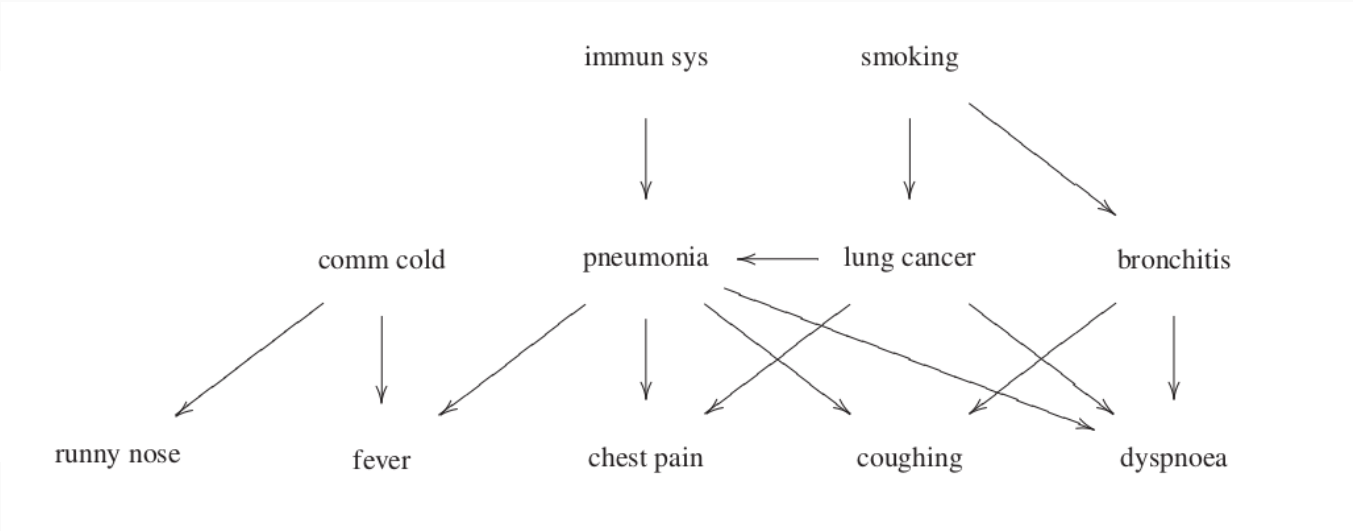
# Tree Augmented Naive (TAN) Bayes Classifiers

- We can see that initially, the **structure** of the DAG is **uknown**.
- Hence, TAN classifiers utilize a modification of the **Chow Liu** algorithm
- This algorithm is used to induce graphical model' structures, with the restriction of a finite number of parental vertices for each node.

# Test Cases

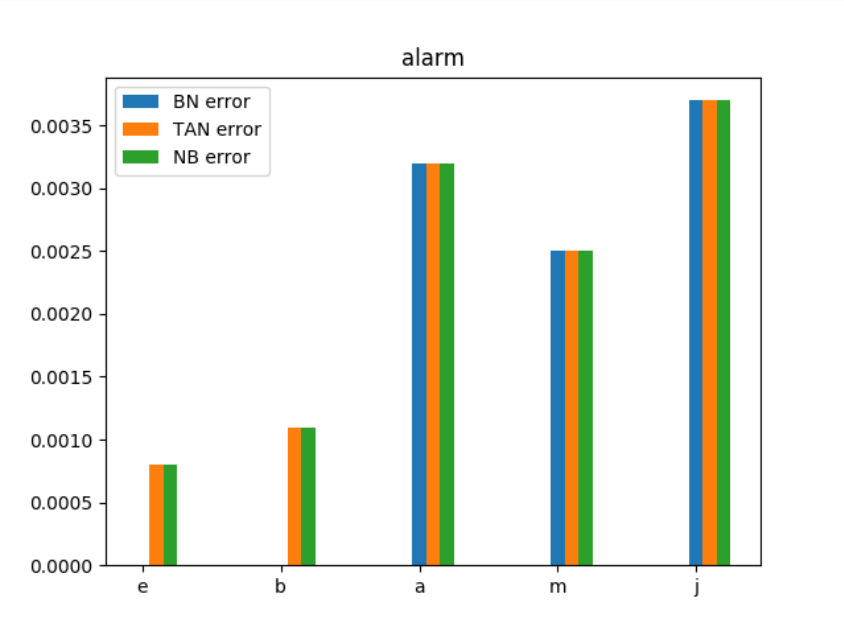
The aforementioned algorithms were tested with the Bayesian Networks whose DAGs a shown below:

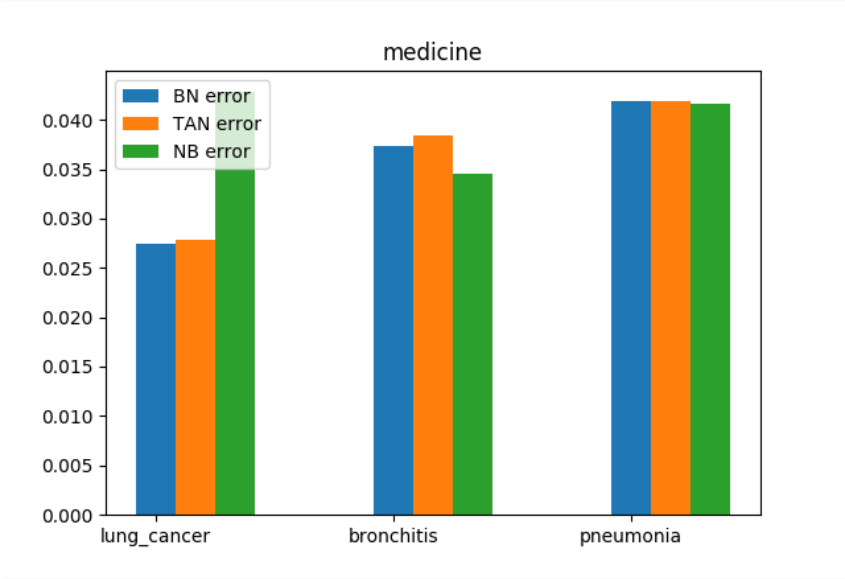


# Classification Error

Below we can see the percentage of the samples that are wrongly classified using the methods that we discussed earlier, as well as the results of the initial networks (BN), for **various labels**.



# Classification Error

- We can see that in the case of the smaller alarm network, both TAN and Naive Bayes classifiers have an efficient performance that is fairly close to the performance of the exact inference to the initial network. This leads us to the conclusion that they have modeled the random variable dependencies well enough in order to achieve low classification error.
- On the other hand, we can also see that when tested to the somewhat more complex medical network, they still manage to approximate random variable dependencies well enough in most cases. However, we see that there is a case where Naive Bayes has significant lower performance.