CS 584 Final Project Proposal

Chih-Feng Yu, A20314374
Karthik Bangalore Mani, A20344597

- **Problem Description**

Learning an optimal Bayesian Network which captures all the dependencies from data is extremely time consuming and is NP-hard. Hence we make the naïve assumption that features are independent given class. This type of a classifier is called Naïve Bayes classifier.

Naïve Bayes classifiers is one of the simple probabilities classifiers that assumes all features are independent (naive) to each other. It applies Bayes' theorem which describes the probability of a conditional event. As the characteristics of Naïve Bayes we learned until now, given the n independent features vector X= $(x_1, \dots, x_n)$, k categories of C, and assuming each feature F_i is conditionally independent to other F_j which $j \neq i$,

$$p(x_i|x_{i+1}, \dots, x_n, C_k) = p(x_i|C_k)$$

For $i = 1, \dots, n-1$, the joint conditional probability model is

$$\begin{aligned} p(C_k|x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) \\ &\propto p(C_k)p(C_k|x_1) \cdots p(C_k|x_n) \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \end{aligned}$$

The resulting classifier is called a naive Bayesian classifier, Or simply naive Bayes. The classifier is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k\epsilon\{1,\dots,K\}}{argmax}\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

However, in the real world, not really all the features are independent to each other. One attribute might be influenced by others which means a feature might not only have just one parent. Therefore, we use another approach called **TAN** (Tree

Augmented Naïve Bayes). Tree Augmented naive Bayes (TAN) is a structure in which a class node points to all Feature nodes , and each feature node can have at most one feature parent.

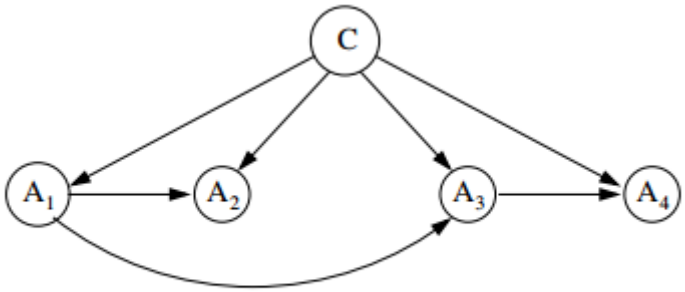


Figure 2: An example of TAN

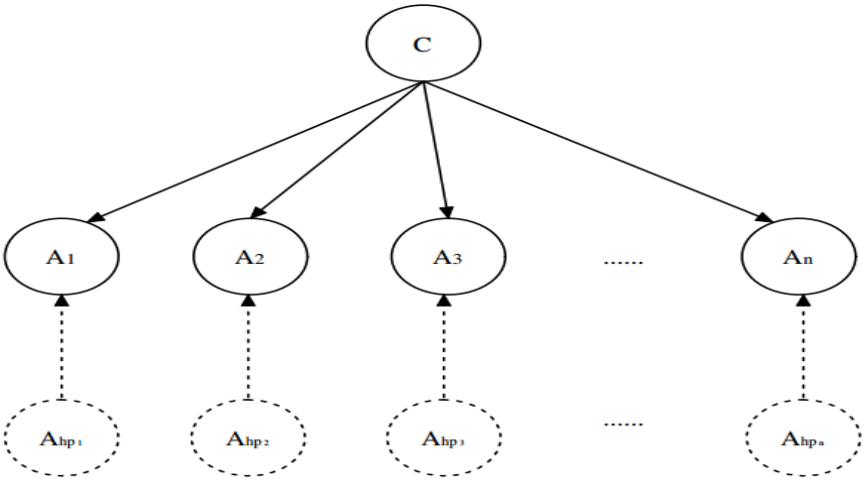
In practice, imposing restrictions on the structures of Bayesian networks, such as TAN, leads to acceptable computational complexity and a considerable improvement over naive Bayes. One main issue in learning TAN is that only one attribute parent is allowed for each attribute, ignoring the influences from other attributes. In addition, in a TAN learning algorithm, structure learning is also unavoidable.

Therefore, the research paper provide a new model called ***Hidden Naive Bayes (HNB)*** to solve this issue.

Therefore, in this project, we will implement and simulate the paper's algorithm to test its performance with some well-chosen datasets.

- ## The methods or algorithms

HNB propose another new feature called **hidden parent** for each attribute, which involving the influence of other attributes. The following figure shows the idea of HNB. C is the class node which is the parent of all attributes ($A_1 \dots A_n$). Each A_i attribute has hidden parent A_{hpi}, $i = 1, 2, \dots , n$.



The joint distribution represented by an HNB is defined as follows.

$$p(A_1, \ldots, A_n, C_k) = p(C_k) \prod_{i=1}^{n} p(A_i | A_{hpi}, C_k)$$

where

$$p(A_i | A_{hpi}, C_k) = \sum_{j=1, j \neq i}^{n} W_{ij} \; p(A_i | A_j, C_k)$$

and $\sum_{j=1, j \neq i}^{n} W_{ij}$ is a mixed of the weighted influences from all other attributes. There is an approach to determining the W_{ij} by using the conditional mutual information between two attributes which is shown as follows.

$$W_{ij} = \frac{I_p(A_i | A_j, C_k)}{\sum_{j=1, j \neq i}^{n} I_p(A_i | A_j, C_k)}$$

where $I_p(A_i | A_j, C_k)$ is the **conditional mutual information**.

The conditional mutual information is given by the equation:

$$I_P(X; Y | Z) = \sum_{x,y,z} P(x, y, z) log \frac{P(x, y | z)}{P(x | z) P(y | z)},$$

The classifier corresponding to HNB on $\mathrm{E} = (a_1, \ldots, a_n)$ is defined as follows.

$$c(E) = \underset{c \epsilon C}{argmax} \; p(c) \prod_{i=1}^{n} p(a_i | a_{hpi}, c)$$

The learning algorithm for HNB:

Algorithm HNB(*D*)
Input: a set *D* of training examples
Output: an hidden naive Bayes for *D*
for each value *c* of *C*
Compute *P(c)* from *D*.
for each pair of attributes A_i and A_j
for each assignment a_i, a_j , and *c* to A_i, A_j , and *C*
Compute P(a_i, a_j |*c*) from *D*
for each pair of attributes A_i and A_j
Compute $I_P(A_i, A_j \mid C)$
for each attribute A_i
Compute $W_i = \sum_{j=1, j \neq i}^{n} I_p(A_i | A_j, C)$
for each attribute A_j and j $\neq$ i
Compute $W_{ij} = \frac{I_p(A_i | A_j, C)}{W_i}$

The training time complexity of HNB is $O(tn^2+kn^2v^2)$, where t is the number of training examples, n is the number of attributes, k is the number of classes, and v is the average number of values for an attribute.

- ## Data usage

We will use the datasets recommended by the paper. It is suggested the datasets from the UCI repository. For handling the missing values, we are planning to use sk-learn's **sklearn.preprocessing.Imputer** function.¶