# CSCI 446 Artificial Intelligence Project 3 Design Report

Roy Smart Nevin Leh Brian Marsh

November 4, 2016

## 1 Introduction

Introduce Machine Learning Algorithm (MLA)!!!! Explain classification problem!!!

## 2 Datasets

### 2.1 Dataset Representation

We will define a *datum* to be a vector consisting of the class as the zeroth element and the associated attributes as the rest of the elements. Classes and attributes will be represented by an integer. Continuous data will be binned before it is inserted into each datum. The resolution of the bins will be a variable that will have to be tuned. Each dataset will be represented as a vector of datums.

### 2.2 Data Imputation

Imputation is the process of approximating missing values in the datasets. To our knowledge there is only one dataset that has real missing values: the Wisconsin Breast Cancer Database. The 1984 United States Congressional Voting Records Database appears to have missing values, but these can actually be interpreted as a stance on a particular issue. Since the breast cancer database has a small proportion of missing values, it is appropriate to simply eliminate datums with missing values. The authors assert that trying to train a MLA with imputed values would only create unnecessary bias in the network.

However, it is a common real-world problem to attempt to classify an unknown, incomplete datum, therefore we will perform imputation on the validation datasets. To approximate the missing values we will first try a hot-deck imputation, where missing attribute values of a given datum are constructed by selecting a random member of that datum's class and copying the value of the attribute. If the hot-deck is unsuccessful, we will attempt to fill in the missing values using a regression model developed in *Mathematica*.

### 2.3 Cross-validation

To partition the full datasets into test and training datasets, we will use 10-fold cross validation. This method partitions the data into ten *folds* and uses one fold for testing data and the remaining nine folds for the training dataset. This process is repeated nine more times until every fold has been used as a test dataset. We selected this method because it will allow the convergence measurement described in Section 5.3 to be applied over a larger range, which allows us to measure the rate of convergence for each MLA more accurately.

# 3 Machine Learning Algorithms

## 3.1 $k$-Nearest Neighbors

## 3.2 Naive Bayes

### 3.2.1 Description

*Naive Bayes* uses the principle of Bayesian learning to solve the classification problem. This algorithm is said to be naive because it considers the attributes to be conditionally independent when performing classifications. Under this scheme, the probability distribution of each class $C$ is given the set of attributes $x_1, ..., x_n$ is written by Russell and Norvig as

$$\mathbf{P}(C|x_1, ..., x_n) = \alpha \mathbf{P}(C) \prod_i \mathbf{P}(x_i|C)$$

where, using the maximum likelihood hypothesis, $\mathbf{P}(C)$ prior probability of the class $C$ in the training dataset, $\mathbf{P}(x_i|C)$ is the likelihood of attribute $x_i$ given $C$, and $\alpha$ is a normalization constant. To find $\mathbf{P}(C)$ the we simply say it is equal to the proportion of $C$ observed in the training dataset. Likewise we can calculate the likelihood using

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

where the probabilities are calculated using the number of occurrences in the training dataset. In this way, naive Bayes can make predictions using only the probability ratios constructed from the training dataset. The probability distribution in the naive Bayes algorithm can be viewed as a Bayesian network, shown in Figure 1 where each attribute is only connected to the class, and there are no relationships between attributes.

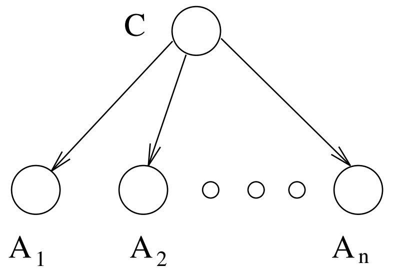

**Figure 1:** *An example of a Bayesian network for naive Bayes. The root $C$ is the class, the leaves $A_1, ..., A_n$ are the attributes [Friedman et al., 1997]*

### 3.2.2 Design Implications

This algorithm should be very straightforward to implement. During the training phase, we simply loop through the dataset and tally how many times each class takes on every value of each attribute. During the testing phase, the algorithm uses the equations described in Section 3.2.1 to select the most probable class.

## 3.3 TAN

### 3.3.1 Description

*Tree-Augmented Naive Bayes* (TAN) is an extension to the naive Bayes algorithm described in Section 3.2. First described by [Friedman et al., 1997] TAN relaxes the assumption that the attributes are conditionally independent and allows each attribute to depend on only one other attribute. It accomplishes this by constructing a fully-connected, undirected graph out of the attributes in the Bayesian network for the naive Bayes algorithm (Figure 1). TAN then assigns a weight to each connection in the graph using the *mutual information function*, given by

$$I_P(\mathbf{X};\mathbf{Y}) = \sum_{\mathbf{x},\mathbf{y}} P(\mathbf{x},\mathbf{y}) \log \left( \frac{P(\mathbf{x},\mathbf{y})}{P(\mathbf{x})P(\mathbf{y})} \right).$$

Applying the this weight to the edges allows us to construct a minimum spanning tree of the graph. We can then make the spanning tree directed by choosing a root node and setting the direction of all edges to be outward from it. The result is exemplified in Figure 2.

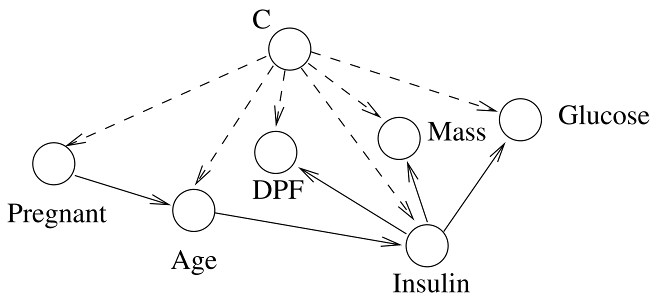

**Figure 2:** *An example of the directed minimum spanning tree produced by TAN. Dashed lines represent the original naive Bayes representation and the solid lines represent the spanning tree [Friedman et al., 1997].*

#### 3.3.2 Design Implications

### 3.4 ID3

## 4 Software Architecture

## 5 Experiment Design

### 5.1 Algorithm Accuracy

#### 5.1.1 Precision

#### 5.1.2 Recall

#### 5.1.3 Confusion

### 5.2 Algorithm Time-Complexity

### 5.3 Algorithm Convergence

$$H(S) = \sum_{i=1}^{n} (-p_i \log_2 p_i)$$

## References

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29(2):131–163.