CSCI 446 Artificial Intelligence Project 2 Final Report

Roy Smart   Nevin Leh   Brian Marsh

November 21, 2016

Abstract

1 Introduction

Define MLA acronym here!!!!

In a broad sense, machine learning refers to efforts to give computers the ability to learn without explicit instructions and encompasses a wide range of problems. Classification in the realm of machine learning is a problem that can be solved using several algorithms, but the effectiveness of the algorithm depends greatly upon the specific dataset involved. We implemented four algorithms to attempt to classify new data points into sets based upon previous information gained through âĂIJtraining.âĂİ The algorithms used are k-Nearest Neighbors, NaÃŕve Bayes, Tree Augmented NaÃŕve Bayes (TAN), and Iterative Dichotomiser 3 (ID3). Additionally, each algorithm is tested with five different datasets: the Wisconsin Breast Cancer Database, the Glass Identification Database, the Iris Plants Database, the Small Soybean Database, and the 1984 United States Congressional Voting Records Database. The effectiveness of each algorithm is measured by the metrics of precision and convergence.

2 Datasets

In this project we are asked to train our MLAs on five different datasets: the Wisconsin Breast Cancer Database, the Glass Identification Database, the Iris Plants Database, the Small Soybean Database, and the 1984 United States Congressional Voting Records Database. Each dataset has a different set of challenges to our MLAs, for example, the soybean dataset is really small, the cancer database is reasonably large, and the attributes in the glass dataset are not very well correlated (as we will soon see).

2.1 Discretization

Two datasets, the iris dataset and the glass dataset have continuous values. We are asked to discretize these dataset to more accurately compare the different MLAs. For our project, we adopted *binning* as our discretization scheme. Binning is a simple method for discretization that defines a certain number of adjacent intervals, known as bins, and then replaces points that fall within each bin by a value representative of that bin.

For the data in this report the number of bins was set to 10. In principle, the ideal number of bins in terms of MLA precision could be any integer larger than one, but through tests we found that the ideal number of bins is proportional to the number of classes in the dataset.

2.2 Cross-Validation

Cross-validation is statistical bootstrapping techinque used to estimate the generality of a statistical model using a dataset independent of the dataset used to create the model. In the context fo machine learning, this means that we will partition the full datasets into training and validation datasets. For this project we are

asked to use the so-called k-fold cross validation technique, where the full dataset is partitioned into training and testing datasets through the use of *folds*. The folds are found by splitting the dataset into k equal sized pieces, and assigning each to be a fold. One fold is selected as the validation set, and the rest of the folds are assigned to the training set. This process is then repeated k times, where each fold is taken to be the validation set once.

To further increase the accuracy of our cross-validation process, we will *stratify* the folds described by k-fold cross-validation technique. Stratification in this context means that each fold will have an equal distribution of the classes in each dataset.. This process makes cross-validation more accurate, because it gives each MLA adequate information on each class for every training dataset. Without stratification, some training datasets could give a MLA zero examples of a particular class, hurting its ability to generalize.

In k-fold cross-validation, k is a free parameter that can be tuned by the user. For this project, we have selected $k = 10$, because it will allow us to perform the convergence tests discussed in Section 7.1 over a larger range.

2.3   Missing Values

Some of the datasets have missing values. This is problematic since most of the MLA's will yield vastly different answers if some data is missing. However, we note that the appearance of missing values in the voting dataset is merely an illusion, missing values in this dataset can signify a stance on a particular issue and this information should be used by our algorithms.

Since the voting set contains no missing values, the only dataset with missing values is the cancer dataset. Since there are so few missing values and since the cancer dataset is so large, we have decided to simply delete any data with missing values from the cancer dataset. Therefore, we have found that data imputation is not necessary for this project.

3 k-Nearest Neighbors

3.1   Training

K-Nearest Neighbors attempts to solve the classification problem by using already-known data points with similar characteristics to make an educated guess of the class. The training for k-Nearest Neighbors involves simply reading in all of the dataset and storing these records as points based on their values.

3.1.1   Constructing Probability Table

3.2   Validation

3.2.1   Value Distance Metric

3.2.2   Determining k and p

4   Naive Bayes

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 [Russel and Norvig, 2010] as

$$\mathbf{P}(C|x_1, ..., x_n) = \alpha \mathbf{P}(C) \prod_i \mathbf{P}(x_i|C) \tag{1}$$

where, using the maximum likelihood hypothesis, $\mathbf{P}(C)$ is the 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 α is a normalization constant. To find

(a)   **(b)**   **(c)**

(d)   **(e)**   **(f)**
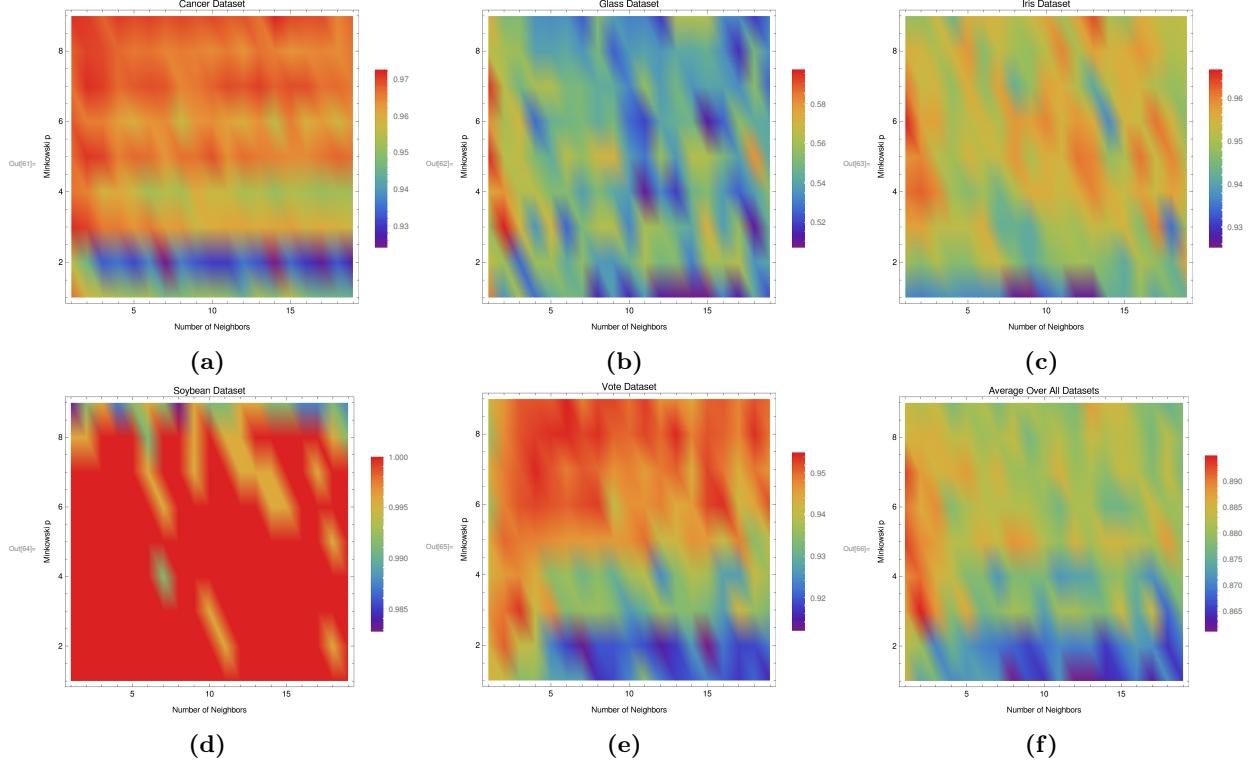
Figure 1

$\mathbf{P}(C)$ the we simply say it is equal to the proportion of C observed in the training dataset. Similarly, we can calculate the likelihood using the definition of conditional probability

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

where the probabilities are calculated using the number of occurrences in the training dataset.

4.1   Training

During the training phase for naive Bayes, the algorithm simply counts the how many times a specific attribute takes on a particular value for each class. From these counts we build the probability table visualized in Figure 2. In Figure 2, we have plotted the class vs. attribute index vs. attribute value vs. probability (in color) for each of the five datasets. These plots are interesting because they serve to visualize the relationships between classes and attribute values.

If we contemplate the plots in Figure 2, we can start to visualize how Naive Bayes works. For the cancer dataset in Figure 2a, we can see how benign (class = 1) results are characterized by low values for every attribute. In the glass dataset, Figure 2b, the low classes (building and vehicle windows) are very well characterized by the attributes, while the miscellaneous glass types (containers, tableware, headlamps) were not dependent upon the attributes. In the iris dataset (Figure 2c), we can see beautiful regions that characterize each species. The soybean dataset in Figure 2d demonstrates a probability table with an interesting geometric structure, while in the voting dataset, Figure 2e, it seems almost as if we can see the party line between Democrats and Republicans.
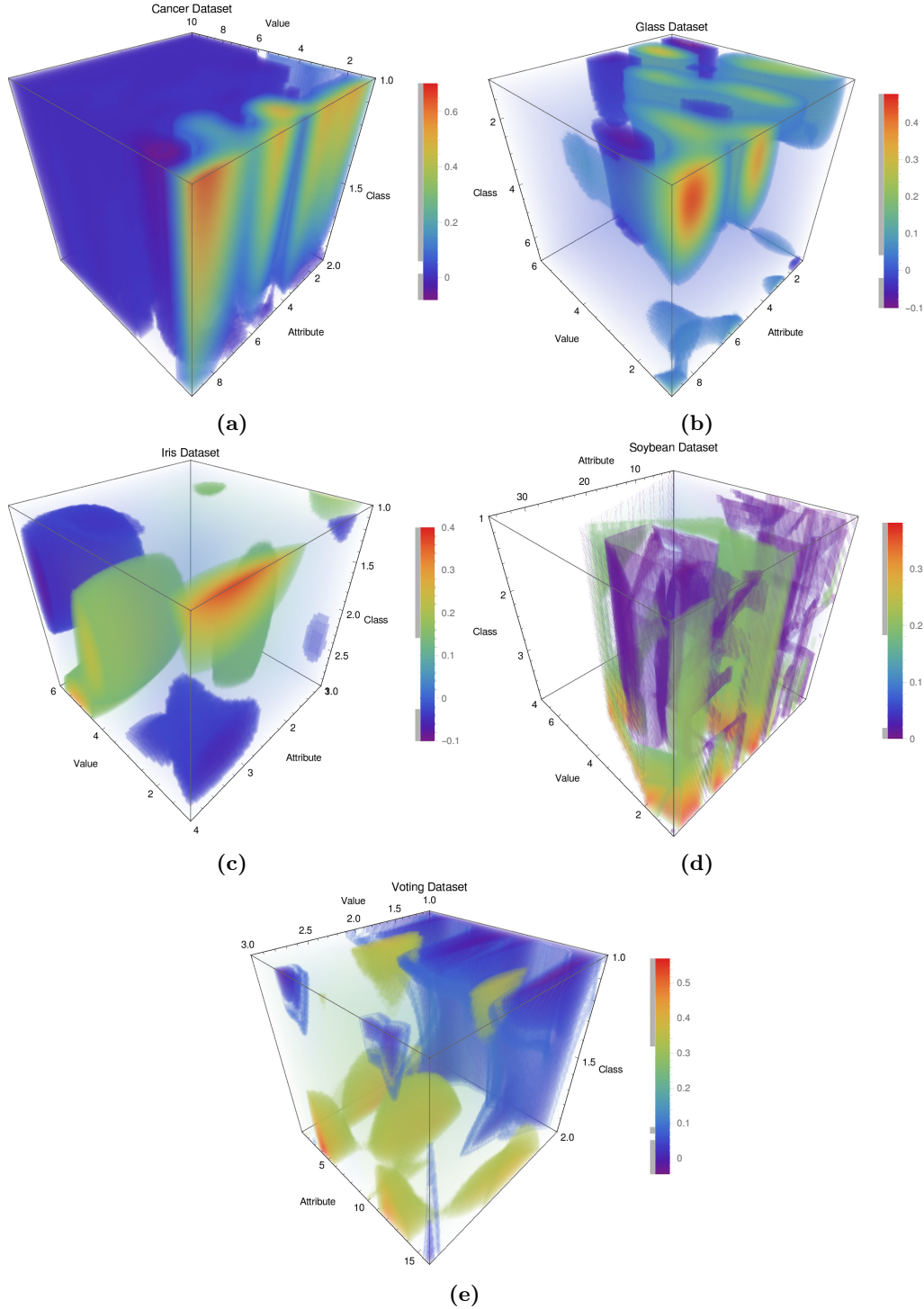
Figure 2: *3D density plots of the naive Bayes probability table for all five datasets. The vertical class axis represents the value of the class, c, the attribute axis denotes the index of an attribute, x, and the value axis represents the vale of that attribute, a. Finally the color is equivalent to the probability of the attribute x having value a, and the class having the value c*

4.2   Validation

To validate our naive Bayes classifier, we simply use the probability table discussed in Section 4.1 and Equation 1 to find the most probable class. To evaluate Equation 1, our implementation looped over the

classes to construct the product of the conditional probabilities. $P(C)$ was calculated by taking the total occurrences of each class and dividing by the total size of the dataset. $P(x_i|C)$ was found by taking the number of times an attribute i with a specified class C had value x_i divided by the total number occurrences of class C;

5   TAN

Tree-Augmented Naive Bayes (TAN) is an extension to the naive Bayes algorithm described in Section 4. Developed 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 dataset. TAN then assigns a weight to each connection in the graph using the *conditional mutual information function*, given by

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

Applying the this weight to the edges allows us to construct a maximum 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.

Using this representation, we can calculate the analog of Equation 1, the probability distribution, using the expression given by [Zheng and Webb, 2010] class, attribute 1 index, attribute 1 value, attribute 2 index, attribute 2 value, count

$$\text{class} = \text{argmax}_C \left[P(C)P(x_r|C) \prod_{x_i,x_j} P(x_j|x_i, C) \right] \tag{2}$$

where x_i is the parent of the attribute x_j in the directed minimum spanning tree.

5.1   Training

To accomplish the TAN algorithm, we will need to construct a probability table similar to the one described in Section 4 on naive Bayes. For this algorithm however, the 4D (class, attribute index, attribute value, count) probability table of naive Bayes needs to be extended to a 6D probability table where the dimensions are: class, attribute 1 index, attribute 1 value, attribute 2 index, attribute 2 value, count, to represent probabilites of the form $P(x_i, , C)$

5.2   Validation

5.2.1   Determining Class Probability Distribution

6   ID3

6.1   Training

6.1.1   Tree Construction

The `id3` method was the main logic behind tree construction. We closely followed the decision tree learning psuedocode in Figure 18.5 in [Russel and Norvig, 2010]. ID3 is an recursive process that generates a short tree by splitting the tree one attribute at a time. The first attribute to be used is the one with the most information gain. Information gain can described as how much entropy the system lost by splitting on an attribute. The equation for entropy is as follows:

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

where p_i is the proportion of the number of datums in class i to the total number of datums in the set S. To use this equation the entropy of the whole system is determined first. Then the sum of the entropy for each new branch is calculated and subtracted from the total entropy. We try to branch on each attribute that has not been used yet. Since the goal is to maximize this difference so the tree will be small and more general, we choose the attribute that resulted in the greatest reduction of entropy. The reduction of entropy is defined as information gain. The rest of the algorithm is just iterating through the tree as it is made and recursively perform the calculation on an attribute that has not been split yet. The tree is finished when all leaf nodes are a single class or when attributes to split on have run out. In this case each leaf node that does not have a class is assigned the most common class in that leaf.

6.1.2 Pruning

We implemented reduced error pruning to help make the tree more general. The approach was to split the training set into a smaller training set and a validation set. The tree was then generated as usual using the new training set.

6.2 Validation

7 RESULTS

7.1 Algorithm Convergence
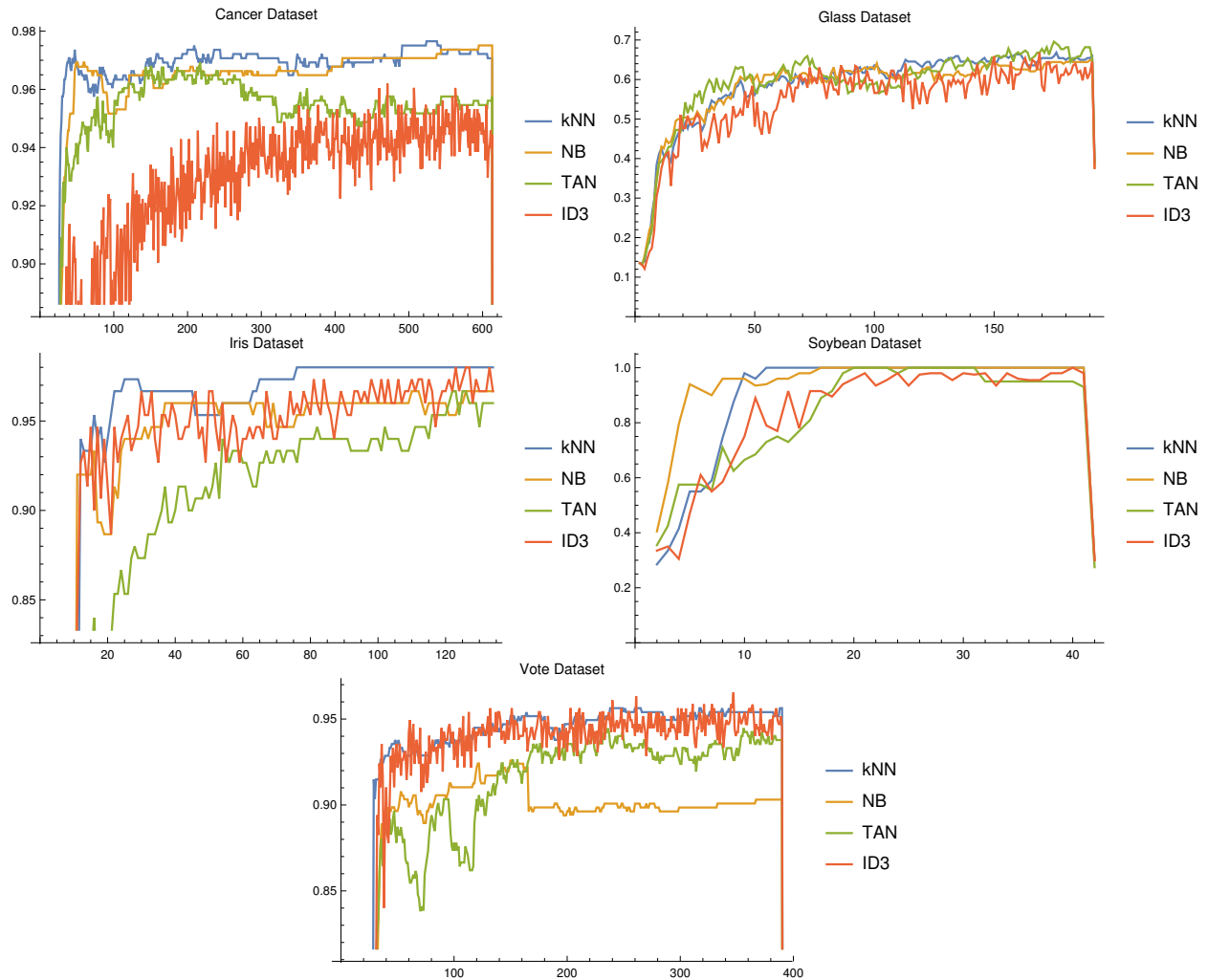
7.2 Algorithm Precision

8 CONCLUSION

Figure 3

REFERENCES

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

[Russel and Norvig, 2010] Russel, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey 07458, 3rd edition.

[Zheng and Webb, 2010] Zheng, F. and Webb, G. I. (2010). *Tree Augmented Naive Bayes*, pages 990–991. Springer US, Boston, MA.