CS6735 Programming Assignment 3

Justin Kamerman 3335272

December 2, 2010

1 Assignment

1. Read the following paper: *Ron Kohavi, Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. KDD-96.*
   You can download it from: http://robotics.stanford.edu/ ronnyk/ronnyk-bib.html

2. Implement the Naïve Bayes tree algorithm using Java. Evaluate your implementation on the data sets in data.zip using 10 times 5-fold cross-validation, and report the average accuracy and standard deviation.

3. Compare it with ID3 and Naïve Bayes in terms of accuracy.

   For breast cancer data see: http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+

   For car data see: http://archive.ics.uci.edu/ml/datasets/Car+Evaluation

   For ecoli data see:http://archive.ics.uci.edu/ml/datasets/Ecoli

   For letter recognition data see: http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

   For mushroom data see: http://archive.ics.uci.edu/ml/datasets/Mushroom

4. **Bonus question:** Change your implementation to output AUC, and compare NBtree with ID3 and Naïve Bayes in terms of AUC.

2 Learning Algorithm

The program implements the NBTree algorithm described in [2]. While the NBTree algorithm is recursive, it has been implemented iteratively using an application stack. This is a personal preference of the student based on past experience with recursive programming causing JVM stack overflow problems.

In the NBTree implementation, a Naïve Bayes classifier is created at every tree node, not just leaf nodes. Non-leaf Naïve Bayes classifiers are used during classification when there is no edge leaving a node corresponding to the attribute value being examined. This is analogous to storing a default classification at each non-leaf node in the Decision Tree implementation of assignment 1 to allow classification of instances for which no path was constructed to a leaf during training.

Missing attribute values are handled by preprocessing data sets, replacing missing attribute value with the most probable value for the attribute in the data set.

3 Data Sets

Program expects data in CSV format with attributes occurring first and classification at the end of each line. All data files have been preprocessed to fit this format.

car.data

- Number of Instances: 1728
- Number of Attributes: 6
- Attribute Values:

buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

- Missing Attribute Values: none
- Class Distribution (number of instances per class)

unacc	1210
acc	384
good	69
v-good	65

ecoli.data

- Number of Instances: 336
- Number of Attributes: 7
- Attribute Values:

mcg	McGeoch's method for signal sequence recognition.
gvh	von Heijne's method for signal sequence recognition.
lip	von Heijne's Signal Peptidase II consensus sequence score. Binary attribute.
chg	Presence of charge on N-terminus of predicted lipoproteins. Binary attribute.
aac	score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
alm1	score of the ALOM membrane spanning region prediction program.
alm2	score of ALOM program after excluding putative cleavable signal regions from the sequence.

- Missing Attribute Values: none.
- Class Distribution (number of instances per class)

cp	143
im	77
pp	52
imU	35
om	20
omL	5
imL	2
imS	2

mushroom.data

- Number of Instances: 8124
- Number of Attributes: 22
- Attribute Information:

cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises?	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
gill-attachment	attached=a, descending=d, free=f, notched=n
gill-spacing	close=c, crowded=w, distant=d
gill-size	broad=b, narrow=n
gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
stalk-shape	enlarging=e, tapering=t
stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil-type	partial=p, universal=u
veil-color	brown=n, orange=o, white=w, yellow=y
ring-number	none=n, one=o, two=t
ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

- Missing Attribute Values: 2480, all for attribute 11.
- Class Distribution:

edible:	4208 (51.8%)
poisonous:	3916 (48.2%)
total:	8124 instances

letter-recognition.data

- Number of Instances: 20000
- Number of Attributes: 17 (Letter category and 16 numeric features)
- Attribute Information:

lettr	capital letter (26 values from A to Z)
x-box	horizontal position of box (integer)
y-box	vertical position of box (integer)
width	width of box (integer)
high	height of box (integer)
onpix	total on pixels (integer)
x-bar	mean x of on pixels in box (integer)
y-bar	mean y of on pixels in box (integer)
x2bar	mean x variance (integer)
y2bar	mean y variance (integer)
xybar	mean x y correlation (integer)
x2ybr	mean of x * x * y (integer)
xy2br	mean of x * y * y (integer)
x-ege	mean edge count left to right (integer)
xegvy	correlation of x-ege with y (integer)
y-ege	mean edge count bottom to top (integer)
yegvx	correlation of y-ege with x (integer)

- Missing Attribute Values: None
- Class Distribution:

789 A	766 B	736 C	805 D	768 E	775 F	773 G
734 H	755 I	747 J	739 K	761 L	792 M	783 N
753 O	803 P	783 Q	758 R	748 S	796 T	813 U
764 V	752 W	787 X	786 Y	734 Z		

breast-cancer.data

- Number of Instances: 699
- Number of Attributes: 10
- Attribute Information: (class attribute has been moved to last column)

Clump Thickness	1 - 10
Uniformity of Cell Size	1 - 10
Uniformity of Cell Shape	1 - 10
Marginal Adhesion	1 - 10
Single Epithelial Cell Size	1 - 10
Bare Nuclei	1 - 10
Bland Chromatin	1 - 10
Normal Nucleoli	1 - 10
Mitoses	1 - 10

- Missing attribute values: 16
- Class distribution: (2 for benign, 4 for malignant)

Benign	458 (65.5%)
Malignant	241 (34.5%)

4 Program Design

The NBTree [2] algorithm is implemented by a Java program. The only external dependency is on the Apache commons-cli library for parsing commend line options. To that end, the program is operated from the command line, taking options listed in table 1.

The program parses a data file, assumed to contain a set of training instances on each line. Each training instance line is a comma-separated list of attribute values terminated by a classification or target value. Of the training data supplied for the assignment, some had to be preprocessed to fit the expected format.

The program then executes a number of iterations (default is 10) over the entire data set. During each iteration, the data set is folded (default is 5 times), the training set used to create an NBTree. The instances in the test set are then evaluated by the NBTree. The accuracy of the classifications is recorded for each

Option	Description
-d	Generate Graphviz DOT output.
-f arg	Path of data file
-i arg	Number of iterations to perform. Default is 10
-n arg	A comma-separated list of attribute names, matching the order in which they appear in the data file
-o arg	Number of folds to create in the training data during. Default is 5.
-h	Print help message

Table 1: Command line options

test set evaluated. After the final iteration is complete, the mean accuracy and standard deviation are calculated and output by the program.

The program implements various mechanism to facilitate debugging. Throughout the code, log statements have been added using the Java logging framework. The logging output is controlled for individual classes via the *logging.properties* file which the program read on startup. In addition to logging, code was added to generate Graphviz DOT [1] output representing the NBTree created. This output can be rendered using Graphviz dot tool.

The implementation classes and their relationships are represented in a UML class diagram in Figure 1. Following is a brief description of each class:

- **Attribute** represents an attribute's name and value.
- **AttributeSet** is a collection of attributes. It has convenience functions to determine attribute value range and probabilities of discrete values.
- **Classification** represents an instance's classification or the value of the target function.
- **ClassificationSet** is a collection of classifications. It has convenience functions to determine classification value range and probabilities of each classification. These functions are used extensively during the entropy calculation.
- **Classifier** is an interface implemented by any class which can take an instance and generate a classification for that instance. The interface is implemented by the *NBTree* and *NaiveBayesClassifier*.
- **Edge** represents an edge in an *NBTree*. Each Edge is associated with an attribute value.

Figure 1: UML class diagram

- **Evaluation** is a helper class for capturing test run accuracies and calculating final mean and standard deviation values.
- **Fold** is an encapsulation of a training and a test set.
- **Instance** represents a collection of attributes and their classification. In the case of training data, we use the classification to select the target hypothesis. In the case of test data, the classification is used to validate output of the target hypothesis.
- **InstanceSet** is a collection of *Instances*. This is the class that parses the data file and generates individual Instances. Functions exist to calculate entropy and information gain, create a subset of instance based on a particular attribute's value and fold the set of instances to create test and training sets of instances.
- **LogFormatter** is a helper class to format log messages.
- **NBTreeMain** is the class which is started from the command line. It parses command line options and starts the test iterations.
- **NaiveBayes** creates a Naïve Bayes Classifier given a training data *InstanceSet*. It calculates marginal and conditional probabilities and populates a probability matrix within the *NaiveBayesClassifier*. It uses the m-estimation technique to estimate conditional probabilities.
- **NaiveBayesClassifier** classifies an *Instance* using the Naïve Bayes Classifier, based on a matrix of marginal and conditional probabilities which are calculated from a previously seen training set.
- **NBTree** creates an *NBTreeClassifier* given a training data *InstanceSet*.
- **NBTreeClassifier** classifies an *Instance* using the encapsulated tree structure and associated *NaiveBayesClassifier* instances created during training.
- **Node** represents a node in an *NBTree*. Each node is associated with either a particular attribute or classification depending on whether the node is an internal or leaf respectively.
- **TreeIterator** is a helper class for performing a depth-first traversal of the *NBTree*.

5 Results

The results of processing each data file using the NBTree implementation are summarized in Table 2. In general, learning time was significantly higher than both the ID3 and Naïve Bayes learning algorithms. Intuitively, this was expected because of the large number of Naïve Bayes classifiers that are created and evaluated during the utility calculation.

All tests were run on a personal computer with an AMD Athlon 64 2GHz Processor, 1GB RAM, running a 32 bit Linux 2.6.31 kernel. The Java Virtual Machine used was version 1.6.0-18.

Data Set	**Naïve Bayes**	**ID3**	**NBTree 5%**	**NBTree 3%**
car	0.841449	0.938377	0.882783	0.925797
ecoli	0.684776	0.384776	0.674328	0.668657
mushroom	0.952007	1.000000	0.951850	0.996921
letter-recognition	0.737670	0.760870	0.819500	
breast-cancer-wisconsin	0.972662	0.945899	0.972806	0.971223

Table 2: Comparison of Naïve Bayes, ID3, and NBTree accuracy over benchmark data sets. NBTree results for both 3% and 5% split utility thresholds are shown

car.data

Using a split utility threshold of 5%, the car evaluation was run through 10 iterations of 5-fold cross-validation. Calculated **mean accuracy** was 0.882783 and **standard deviation** 0.047129. The NBTree split utility threshold of 5% relative improvement in accuracy, did not justify a split on any attribute. Essentially, this is a Naïve Bayes classifier and evaluation results are within a single standard deviation of the results of the previous assignment.

The data set was re-evaluated using a split utility threshold of 3%, resulting in improved **mean accuracy** of 0.925797 and **standard deviation** 0.01811. The lower split threshold drove the formation of the NBTree structure shown in Figure 2.

ecoli.data

Using a split utility threshold of 5%, the ecoli evaluation was run through 10 iterations of 5-fold cross-validation. Calculated **mean accuracy** was 0.674328

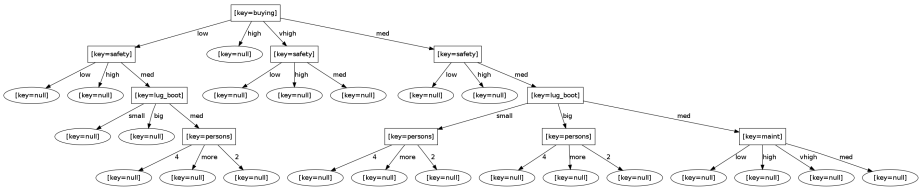

Figure 2: NBTree visualization generated for car data set with 3% split utility threshold

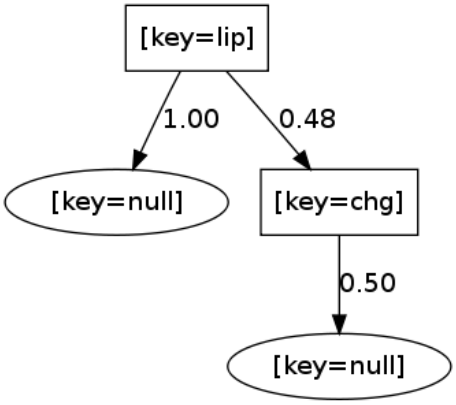

Figure 3: NBTree visualization generated for ecoli data set with 5% utility

and **standard deviation** 0.057782. Although the NBTree split utility threshold of 5% relative improvement in accuracy, justified a split, the resulting tree did not perform significantly better than the Naïve Bayes classifier in the previous assignment. The NBTree structure produced for one of cross-validation folds is shown in Figure 3.

The data set was re-evaluated using a split utility threshold of 3%. The adjusted threshold did not see a significant change in accuracy, yielding **mean accuracy** of 0.668657 and **standard deviation** 0.061466.

mushroom.data

Using a split utility threshold of 5%, the mushroom evaluation was run through a single iteration of 5-fold cross-validation. The full 10 iterations required was not completed due to time constraints, a single fold taking in excess of three hours to complete. Calculated **mean accuracy** was 0.95185 and **standard deviation**

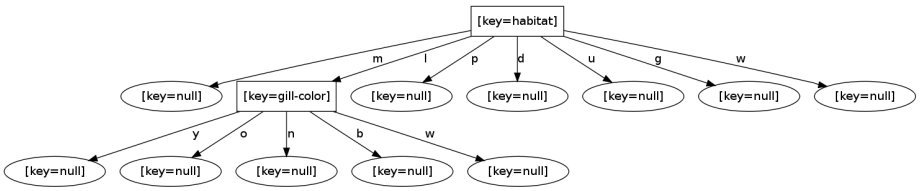

Figure 4: NBTree visualization generated for mushroom data set with 3% utility

0.00856. The NBTree split utility threshold of 5% relative improvement in accuracy, did not justify a split on any attribute. Essentially, this is a Naïve Bayes classifier and evaluation results are within a single standard deviation of the results of the previous assignment.

The data set was re-evaluated using a split utility threshold of 3%, yielding an **accuracy** of 0.996921. Time constraints only permitted evaluation of a single fold so it it is impossible to assess the accuracy variance. The lower split threshold drove the formation of the NBTree structure shown in Figure 4.

letter-recognition.data

The letter-recognition evaluation ran a single fold of 5-fold cross-validation. The full 10 iterations of 5-fold cross validation required was not completed due to time constraints, a single fold taking in excess of twelve hours to complete. **Accuracy** for the single fold was 0.819500. It is difficult to say so with conviction given the sparse evaluation results, but this data set seems to be the only one tested to show a possible improvement over ID3 or Naïve Bayes. Perhaps it is not co-incidental that this data set resulted in the largest tree structure. The tree produced for one of the cross-validation folds is shown in Figure 5.

Due to time constraints no attempt was made to investigate the effects of adjusting the split utility threshold.

breast-cancer.data

The breast-cancer evaluation ran 10 iterations of 5-fold cross validation. Calculated **mean accuracy** was 0.972806 and **standard deviation** 0.012568. The NBTree split utility threshold of 5% relative improvement in accuracy, did not justify a split on any attribute. Essentially, this is a Naïve Bayes classifier and evaluation results are within a single standard deviation of the results of the

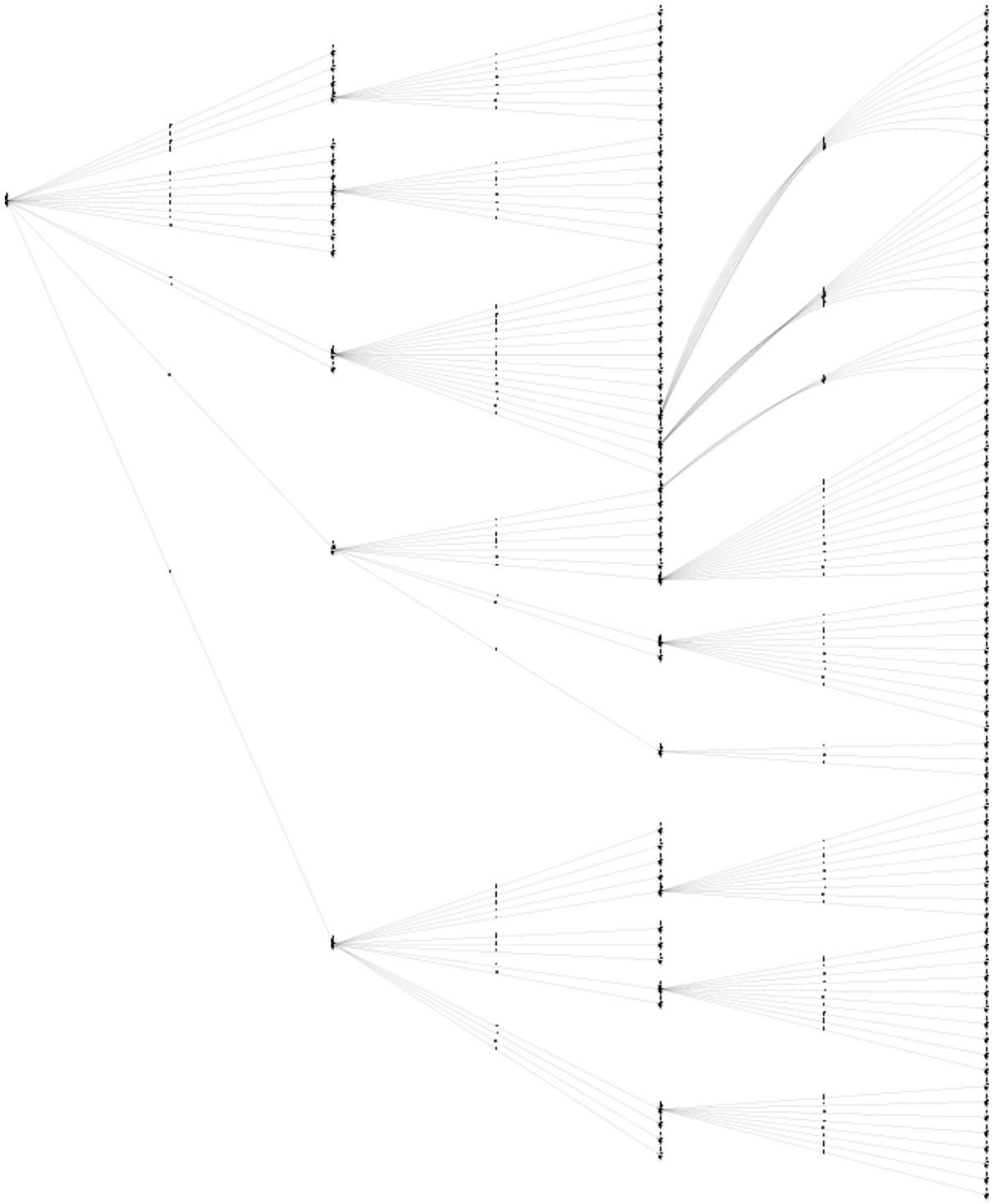

Figure 5: NBTree visualization generated for letter-recognition data set with 5% split utility threshold

previous assignment.

The data set was re-evaluated using split utility thresholds of 3%, 2%, and 1%.

Bibliography

[1] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - open source graph drawing tools. *Graph Drawing*, pages 483–484, 2001.

[2] Ron Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *Second International Conference on Knowledge Discovery and Data Mining*, 1996.