Exercise 3

AMTH/CPSC 445a/545a - Fall Semester 2016

October 7, 2017

Compress your solutions into a single zip file titled `<lastname and initials>_assignment3.zip`, e.g. for a student named Tom Marvolo Riddle, `riddletm_assignment3.zip`. Include a single PDF titled `<lastname>_assignment3.pdf` and any MATLAB or Python scripts specified.

Please include your name in the header of all submitted files (on every page of the PDF) and in a comment header in each of your scripts.

Your homework should be submitted to Canvas before Friday, October 27, 2017 at 5:00 PM.

Programming assignments should use built-in functions in MATLAB or Python; In general, Python implementations may use the `scipy` stack [1]; however, exercises are designed to emphasize the nuances of data mining algorithms - if a function exists that solves an entire problem (either in the MATLAB standard library or in the `scipy` stack), please consult with the TA before using it.

Problem 1

1. Given N data points and a real-valued numerical attribute, suggest an algorithm with complexity $O(N \log N)$ for finding the best binary split of this attribute w.r.t. the Gini index. Justify the complexity of the proposed algorithm.

2. Sketch a full decision tree for the parity function over four binary attributes: A, B, C, and D (e.g., the class is $+$ when the number of attributes that are 1 is even, and it is $-$ when this number is odd). The tree should consider a uniform distribution of all possible combinations for these attributes (i.e., $0000, 0001, 0010, 0011, \ldots$). Can this tree be pruned without degrading its classification performances?

3. Consider the following dataset:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Build a decision tree for deciding whether to play tennis or not based on the weather today.

- Your tree construction should be based on Gini index and multiway splits
- Provide a sketch of the constructed tree in the submitted PDF
- Provide justifications for the choice of each attribute in the construction process.

Problem 2

1. Give an example of binary classification with two binary attributes where Gini index prefers one attribute for a (binary) split while information gain prefers the other attribute (*hint:* you can use duplicate entries of each combination of values). Explain how such an example can occur, even though we saw in class that both entropy and Gini index increase and decrease monotonically in the same regions (i.e., $[0, 0.5]$ and $[0.5, 1]$ correspondingly) for binary classification.

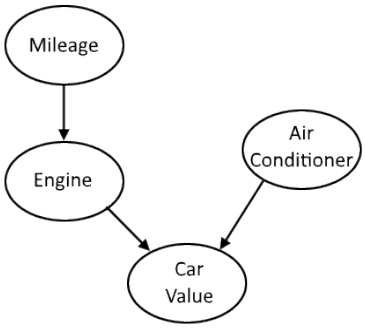
2. Consider the setting of $k \geq 2$ classes, nominal attributes, and multiway splits. Prove that information gain of a split is always positive (i.e., entropy never increases by splitting a node). *Hint #1:* You can use Jensen's inequality for the log function to get $\sum_k a_k \log(b_k)/\sum_k a_k \leq \log(\sum_k a_k b_k/\sum_k a_k)$. *Hint #2:* think in terms of joint and conditional probabilities.

Problem 3

1. Consider the following aggregated dataset with binary attributes:

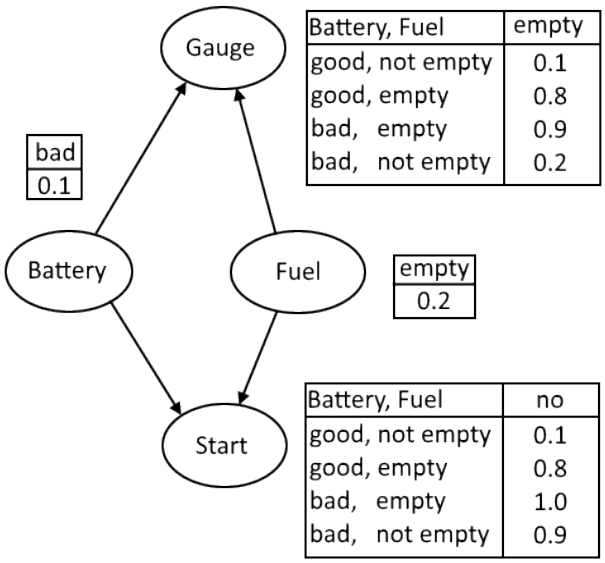
Mileage	Engine	Air Conditioner	frequency(Car Value = Hi)	frequency(Car Value = Low)
Hi	Good	Working	3	4
Hi	Good	Broken	1	2
Hi	Bad	Working	1	5
Hi	Bad	Broken	0	4
Lo	Good	Working	9	0
Lo	Good	Broken	5	1
Lo	Bad	Working	1	2
Lo	Bad	Broken	0	2

- Complete the CDTs for the following Bayesian belief network and add them to the submitted PDF:



- Using this network, compute $\Pr[E = \text{Bad}, AC = \text{Broken}]$ and add the detailed computation to the submitted PDF

2. Consider the following Bayesian belief network and add detailed computations of the following probabilities to the submitted PDF:



- $\Pr[B = \text{good}, F = \text{empty}, G = \text{empty}, S = \text{yes}]$
- $\Pr[B = \text{bad}, F = \text{empty}, G = \text{not empty}, S = \text{no}]$
- The probability that the car starts given that the battery is bad

Problem 4

For this problem you will implement Hunt's decision tree algorithm for categorical data. For each split, your implementation should select the best attribute based on Information Gain, and perform a multi-way split with one child node for each of the possible attribute values (see the lecture slides for a description of Hunt's algorithm and Information Gain). If a leaf node contains more than one class, use majority voting to set the class of the leaf node. If there is a tie in majority voting, choose the smallest class value from those in the tie. If multiple attributes have the same Information Gain, split based on the attribute that first appears in the attribute data matrix.

Your data comes from a curated manifesto of the passengers of the *Titanic*, a British passenger liner that collided with an iceberg in the wee hours of the morning on 15 April 1912. 1,502 people died out of 2,224 total passengers and crewmen. Can you predict who will live and who will die using Hunt's decision tree algorithm? Two files are given to you: the first, `train.csv`, is a comma-separated values document that contains 891 entries, each row corresponding to an individual on the titanic. The columns in this training set are as follows:

Variable	**Definition**
`PassengerId`	Passenger ID
`survival`	Survival (0 = died, 1 = survived)
`pclass`	Ticket class (1=1st, 2=2nd, 3= 3rd)
`Name`	Passenger Name
`sex`	Sex
`Age`	Age in years
`sibsp`	# of siblings / spouses aboard the Titanic
`parch`	# of parents / children aboard the Titanic
`ticket`	Ticket number
`fare`	Passenger fare
`cabin`	Cabin number
`embarked`	Port of Embarkation

Upon opening this table you'll first see that some attributes are missing for all or part of the data. You'll also notice that some columns are not in integer form. You'll want to do some preprocessing to clean the data.

- Consider dropping columns with greater than 50% of their entries missing (e.g. cabin number). Rationalize why this could be reasonable given the other variables you have and their relationship with these columns
- Drop rows that have missing elements. You should have around $\sim$ 700 entries after dropping these rows.
- Come up with a reasonable integer coding scheme for columns that are not categorical. For instance, you may encode `age` and `fare` by creating number-coded ranges and assigning values to each respective range.
- Code strings as integers. For instance `sex` could be coded as 0(male), 1(female). Port of embarkation could be coded as 1,2,or 3.
- Ignore the name column. Any correlation names have with death rate is spurious; if you find one in your training, you'll be overfitting.

We've prepared preprocessing files that take care of most of this for you in `preprocess4.m` and `preprocess4.py`. You'll have to take what we've done as an example and do a similar discretization for the ticket numbers. You can also probably improve upon our preprocessing to improve your accuracy.

The second file you receive is `test.csv`. This is a similar file as `train.csv`, but the survival column has been removed. We will use your predictive performance over the 418 individuals in `test.csv` to in part determine your final grade.

To get a reasonable benchmark of where your model stands, we have created a performance accuracy assessment that will test your performance predicting over `test.csv`. This may be found linked as `accuracy.html` on the website. On this website, you must submit a two column csv with a first-row header. The first column should be labeled `PassengerId` and contain the PassengerID of your prediction. The second column will be labeled `Survived` and contain the aforementioned binary coding scheme for survival. **Sort your input CSV on the first column such that the 418 test cases are ordered by their PassengerID.**

Remark. Failure to follow these guidelines will result in an inaccurate or bugged classification metric. Do not download the html file. It will not work.

Remark. Do not be discouraged by seemingly modest classification results of 70%. It is a difficult dataset for a decision tree to classify well. World class models score 95% on this dataset, but are much more advanced and overfit to this problem than the decision tree you will implement.

Remark. Consider using programming techniques such as **recursion** or **object oriented programming**. MATLAB users should become intimate with `struct`; Pythonistas can seek similar respite with `class`.

All of your code for this problem should be in a single file named either `lastname_script4.py` or `lastname_script4.m` depending on if you're using Python or MATLAB.

Your solution will be graded by testing your code against the test set. Additionally, it will be tested against testing and training data sets from disparate sources, possibly with different numbers of classes, different numbers of attributes, than the titanic data set. Moreover, your code should work in general, not just on the titanic data set.

For MATLAB or Python write functions with the following calling sequences:

$$tr = \text{tree\_train}(c, nc, x, nx)$$

where

- c is an $m \times 1$ vector of integers corresponding to the class labels for each of the m samples,
- nc is the number of classes, that is, $c(i) \in \{1, \ldots, nc\}$ for $i = 1, \ldots, m$.
- x is an $m \times n$ matrix of integers corresponding to the n categorical attributes for each of the m training samples,
- nx is an $1 \times n$ vector corresponding to the range of each of the n attributes, that is, $x(i, j) \in \{1, \ldots, nx(j)\}$ for all $i = 1, \ldots, m$, and
- tr encodes the tree structure for usage in tree_classify. You may choose the structure of tr.

Second, you should implement a classification algorithm with calling sequence

$$b = \text{tree\_classify}(y, tr)$$

where

- y is an $k \times n$ matrix of integers corresponding to the n categorical attributes for each of the k testing samples.
- tr is the tree structure output by your function train_tree, and
- b is an $k \times 1$ vector of integers corresponding to the predicted class of the k testing samples by your decision tree.

If you're using MATLAB, complete the exercise only using the core MATLAB language. If you're using Python, complete the exercise only using the core Python language and the SciPy stack. If you have concerns about a particular function or package, please contact the TA.

Problem 5

For this problem you will use an implementation of CART to build a decision tree model for a leaf data set. You will evaluate your model using several different methods of cross validation. In particular, you should evaluate you model using leave-one-out cross validation, 2-fold cross validation, and 17-fold cross validation. Each cross validation method will make a total of 340 predictions (one for each sample). For example, for leave-one-out cross validation you should make 340 splits (corresponding to the number of samples) such that each sample in excluded once from training, and its predicted value is determined by this split. For each cross validation method, summarize your predictions in a 30×30 confusion matrix (where 30 is the number of classes).

Plot each of the resulting confusion matrices (using `imagesc` in MATLAB or `imshow` in Python) and include the plots in your assignment PDF. Additionally, the classification accuracy for each method should be computed and included in your PDF. Finally, construct a decision tree on the entire dataset and visualize the resutling tree. The method of visualization is detailed below. Save the resulting image as a PNG, and include the PNG file in your assignment ZIP file (because the resulting tree may be very large, there is no need to include this image in your PDF).

Your code should be contained in a single file based on the template `lastname_script5.m` or `lastname_script5.py` depending on if you're using MATLAB or Python. The template script will load preprocessed data `leaf.mat`. The data consists of an m-dimensional vector c of classes represented as positive integer (1-30), and an $m \times n$ floating point attribute matrix. A key for the classes and a description of the attributes is provided in `leaf_key.txt`.

If you're using MATLAB, complete the exercise only using the core MATLAB language. The function `fitctree` fits a decision tree based on the CART algorithm. The model the trained by

$$M = \text{fitctree}(x, c)$$

and classification is performed by

$$c\_hat = \text{predict}(M, y).$$

The decision tree M can be visualized in MATLAB by

$$\text{view}(M, 'mode', 'graph')$$

If you're using Python, you will need to install several packages. First, install `scikit-learn` for your Python distribution. For example, the package can be installed using pip by `pip install -U scikit-learn`. for additional information. Second, install `pydotplus` for your Python distribution. For example, the package can be installed using pip by `pip install -U pydotplus`. Third, install the GraphViz binaries. For example, the binaries can be installed using yum by `yum install graphviz`. You will need to determine the appropriate way to install these packages on your system. For additional information, see scikit-learn.org/stable/install.html. After installing the appropriate packages, complete the exercise only using the core Python programming language and the following functions as needed:

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from scipy.io import loadmat as load
from numpy import argsort, reshape, transpose, array, zeros
from matplotlib.pyplot import imshow, xlabel, ylabel, title, figure, savefig
from numpy.random import permutation, seed
from pydotplus import graph_from_dot_data
```

The function `DecisionTreeClassifier` can be used to train a CART decision tree by

$$M = \text{DecisionTreeClassifier}()$$

$$M = M.\text{fit}(x, c)$$

and then classification can be performed by

$$c\_hat = M.\text{predict}(y).$$

The tree can be visualized by

$$\text{dot\_data} = \text{export\_graphviz}(Md, \text{out\_file} = \text{None})$$

$$\text{graph} = \text{graph\_from\_dot\_data}(\text{dot\_data})$$

$$\text{graph.write\_pdf}('\text{cart\_tree.png}', \text{f='png'})$$

For additional information see scikit-learn.org/stable/modules/tree.html.

Problem 6

For this problem, you will implement a Naïve Bayesian classifier on a seed data set. Your implementation should use the Laplace Correction to avoid issues with zero probabilities. See the lecture slides for a description of Naïve Bayesian classification and the Laplace Correction.

All your code should be in a single file based off the template `lastname_script6.m` or `lastname_script6.py` depending on if you're using MATLAB or Python. The template will load the preprocessed data `seed_data.mat` which consists of an m-dimensional vector of positive integers c (class labels), an integer nc (the number of classes), and an $m \times n$ floating point attribute matrix. The file `seed_data.txt` documents the classes and attributes of this data set.

For both MATLAB and Python your functions should have calling sequence

$$pr = \text{naivebayes\_train}(c, nc, x, nk)$$

where c is an $m \times 1$ vector of positive integer class labels, nc is the integer number of classes, that is, $c(i) \in \{1, \ldots, nc\}$ for all $i = 1, \ldots, m$, x is an $m \times n$ attribute matrix with real values, and nk is the number of histogram bins to use. Moreover, your implementation should represent the probability distributions as histograms with nk bins of equal width. Second, you should implement a classification function

$$c\_hat = \text{naivebayes\_classify}(pr, y)$$

which takes your trained Naïve Bayes model pr as well as a $k \times n$ attribute matrix y, and returns a $k \times 1$ dimensional vector c_hat with the predicted class labels represented as positive integers.

After implementing the Naïve Bayes model, use $nk = 6$ bins to perform leave-one-out cross validation on the data with m splits (one for each data point). Compute the resulting 3×3 confusion matrix. Since the matrix is small, rather that creating a image plot, simply include a print out of the numerical matrix in your PDF. Also compute the classification accuracy of your model, and include this number in your PDF.

Finally, train your Naïve Bayes model on the entire data set (c, x) using bin parameter $nk = 6$, and perform the following visualization. Select two attributes x_i and compute

$$P(x_i|c_j)P(c_j).$$

which is proportional the posterior distribution of x_i given c_j, for $j = 1, \ldots, nc$. Since $nk = 6$, your posterior distribution (up to normalization) will consist of 6 numbers corresponding to the 6 bins in your histogram representation. To visualize the distributions, create a line plot of the the center of each of your nk bins on the x-axis, and the posterior distribution $P(x_i|c_j)P(c_j)$ on the $y - axis$. Add a legend labeling the class c_j of each line. Moreover, your should create two figure with three lines each approximating the posterior distribution (up to normalization) of your chosen attributes given each of the three classes.

If you're using MATLAB, complete the exercise only using the core MATLAB language. If you're using Python, complete the exercise only using the core Python language and the SciPy stack. If you have concerns about a particular function or package, please contact the TA.

References

[1] "The scipy stack specification." [Online]. Available: https://www.scipy.org/stackspec.html