HOMEWORK 2

Alexander Rusnak
9078074888
atrusnak@wisc.edu

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. You can choose any programming language (i.e. python, R, or MATLAB), as long as you implement the algorithm from scratch (e.g. do not use sklearn on questions 1 to 7 in section 2). Please check Piazza for updates about the homework.

1 A Simplified Decision Tree

You are to implement a decision-tree learner for classification. To simplify your work, this will not be a general purpose decision tree. Instead, your program can assume that

- each item has two continuous features $\mathbf{x} \in \mathbb{R}^2$
- the class label is binary and encoded as $y \in \{0, 1\}$
- data files are in plaintext with one labeled item per line, separated by whitespace:

$$x_{11} \quad x_{12} \quad y_1$$
$$\ldots$$
$$x_{n1} \quad x_{n2} \quad y_n$$

Your program should implement a decision tree learner according to the following guidelines:

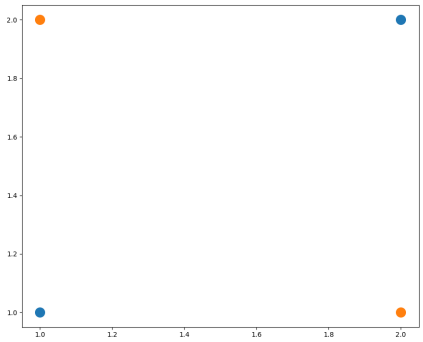
- Candidate splits (j, c) for numeric features should use a threshold c in feature dimension j in the form of $x_{\cdot j} \geq c$.
- c should be on values of that dimension present in the training data; i.e. the threshold is on training points, not in between training points. You may enumerate all features, and for each feature, use all possible values for that dimension.
- You may skip those candidate splits with zero split information (i.e. the entropy of the split), and continue the enumeration.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using information gain ratio. If there is a tie you may break it arbitrarily.
- The stopping criteria (for making a node into a leaf) are that
  - the node is empty, or
  - all splits have zero gain ratio (if the entropy of the split is non-zero), or
  - the entropy of any candidates split is zero
- To simplify, whenever there is no majority class in a leaf, let it predict $y = 1$.

2 Questions

1. (Our algorithm stops at pure labels) [10 pts] If a node is not empty but contains training items with the same label, why is it guaranteed to become a leaf? Explain. You may assume that the feature values of these items are not all the same.

   This is guaranteed to become a leaf because any decision boundary will have an InformationGainRatio of zero. This is clear upon inspection of the informationGain formula. If all the class labels are the same then the entropy is 0.

2. (Our algorithm is greedy) [10 pts] Handcraft a small training set where both classes are present but the algorithm refuses to split; instead it makes the root a leaf and stop; Importantly, if we were to manually force a split, the algorithm will happily continue splitting the data set further and produce a deeper tree with zero training error. You should (1) plot your training set, (2) explain why. Hint: you don't need more than a handful of items.



   Here we have points p1 = (1,1) p2 = (1,2) p3 = (2,1) p4 = (2,2)
   p1 and p4 belong to class 1, while p2 and p3 in class 2
   This will not split because any decision boundary will leave 1 point of each class on either side, and it will still be a 50/50 guess (no information gain). However if we force 1 point to split from the other three, the tree would happily split again.

3. (Information gain ratio exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their information gain ratio. If the entropy of the candidate split is zero, please list its mutual information (i.e. information gain). Hint: to get $\log_2(x)$ when your programming language may be using a different base, use `log(x)/log(2)`. Also, please follow the split rule in the first section.

   (Threshold, axis) InfoGainRatio: x.x

   (0.0, 0) InfoGainRatio: 0.0
   (-2.0, 1) InfoGainRatio: 0.0
   (-1.0, 1) InfoGainRatio: 0.10051807676021828
   (5.0, 1) InfoGainRatio: 0.11124029586339795
   (6.0, 1) InfoGainRatio: 0.23609960614360798

4. (The king of interpretability) [10 pts] Decision tree is not the most accurate classifier in general. However, it persists. This is largely due to its rumored interpretability: a data scientist can easily explain a tree to a non-data scientist. Build a tree from D3leaves.txt. Then manually convert your tree to a set of logic rules. Show the tree[1] and the rules.

[1] When we say show the tree, we mean either the standard computer science tree view, or some crude plaintext representation of the tree – as long as you explain the format. When we say visualize the tree, we mean a plot in the 2D $\mathbf{x}$ space that shows how the tree will classify any points.

Note: in order to obtain any useful tree, my algorithm was modified so points are split if they are $> c$ instead of $\geq c$, where c is a threshold on which to split. This was only done for this question as it goes against the implementation instructions, but I believe this is how it was intended for this question.

Tree Format: Parenthesis represent nodes where splitting occurs in the form (threshold, axis). Each new line is a new layer of the tree. Numbers 0 or 1 represent leaf nodes with the corresponding label. Nodes are displayed in order form left to right. So the children of the furthest left node are the two furthest left children on the new line.

(1.0, 0)

---

1 (1.0, 1)

---

1 0

Rules:
if $x_1 > 1$, then $y = 1$
else if $x_2 > 1$, then $y = 1$
else then $y = 0$

5. (Or is it?) [20 pts] For this question only, make sure you DO NOT VISUALIZE the data sets or plot your tree's decision boundary in the 2D $\mathbf{x}$ space. If your code does that, turn it off before proceeding. This is because you want to see your own reaction when trying to interpret a tree. You will get points no matter what your interpretation is. And we will ask you to visualize them in the next question anyway.

   - Build a decision tree on D1.txt. Show it to us in any format (e.g. could be a standard binary tree with nodes and arrows, and denote the rule at each leaf node; or as simple as plaintext output where each line represents a node with appropriate line number pointers to child nodes; whatever is convenient for you). Again, do not visualize the data set or the tree in the $\mathbf{x}$ input space. In real tasks you will not be able to visualize the whole high dimensional input space anyway, so we don't want you to "cheat" here.
     Same tree format as above.
     (0.199725, 1)

     (0.520225, 0) 0

     ---

     1 1

   - Look at your tree in the above format (remember, you should not visualize the 2D dataset or your tree's decision boundary) and try to interpret the decision boundary in human understandable English.
     First split the data on axis 1 with threshold 0.199. Classify points under the threshold as 0. For numbers greater than this threshold, split the data on axis 0 with threshold 0.52. Even though this split technically has information gain, both splits are labeled 1.
   - Build a decision tree on D2.txt. Show it to us.

     (0.532664, 0)

     (0.224236, 1) (0.884695, 1)

     (0.380406, 1) (0.850316, 0) (0.104043, 0) (0.690829, 1)

     (0.420289, 1) (0.66337, 0) (0.035885, 1) 0 (0.104094, 0) (0.917219, 1) (0.233317, 0) (0.534979, 1)

(0.541762, 0) (0.565323, 0) (0.707502, 0) 0 (0.076499, 1) 0 (0.884872, 1) 1 (0.06542, 0) 0 (0.722322, 1) (0.189016, 0) (0.414023, 0) 0

(0.421367, 1) 1 1 0 (0.283131, 1) (0.291518, 1) (0.090408, 1) (0.927522, 0) 1 1 1 1 (0.253723, 0) 1 (0.782804, 1) (0.864128, 1) (0.570223, 1) (0.409387, 0)

1 1 1 1 (0.68746, 0) 0 (0.861578, 0) (0.07836, 1) (0.053702, 1) 0 1 1 (0.194975, 0) 0 (0.093713, 0) 0 (0.417579, 0) 1 1 (0.393227, 0)

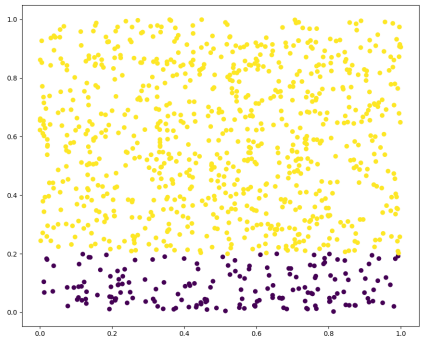
1 1 (0.091239, 1) 1 1 1 1 1 1 1 1 0 1 1 (0.585765, 1) 0

1 1 1 0

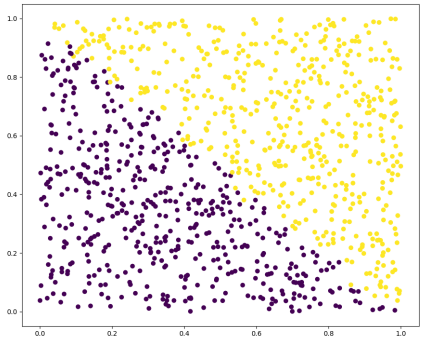
- Try to interpret your D2 decision tree. Is it easy or possible to do so without visualization?
  Very difficult to do without visualization. May be possible, but quite unreasonable

6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:
   - Produce a scatter plot of the data set.
   - Visualize your decision tree's decision boundary (or decision region, or some other ways to clearly visualize how your decision tree will make decisions in the feature space).

   D1 plot



   D2 plot



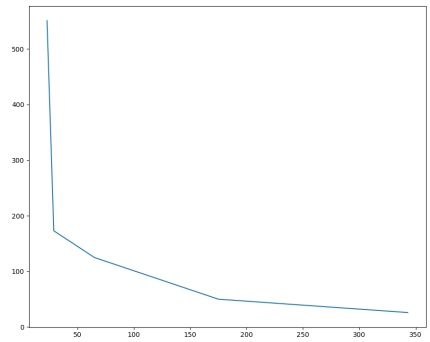
Then discuss why the size of your decision trees on D1 and D2 differ. Relate this to the hypothesis space of our decision tree algorithm.

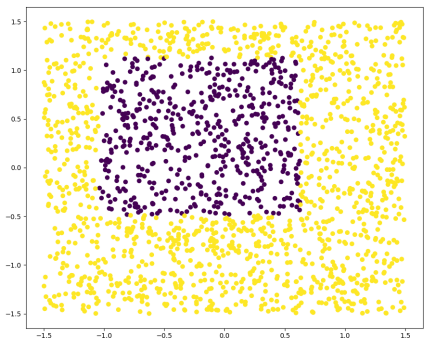
The size of D2 is larger because it's apparent decision boundary is diagonal whereas D1 is horizontal. Our hypothesis space for this decision tree only uses vertical and horizontal boundaries (different thresholds of x1 and x2. Therefore the the model is better suited to a dataset like D1.

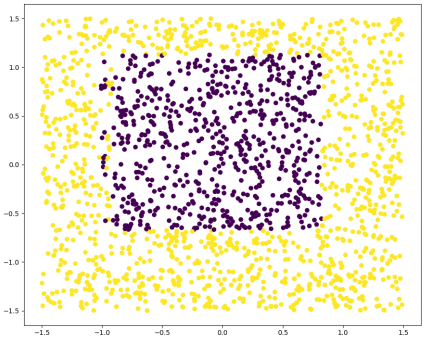
7. (Learning curve) [20 pts] We provide a data set Dbig.txt with 10000 labeled items. Caution: Dbig.txt is sorted.

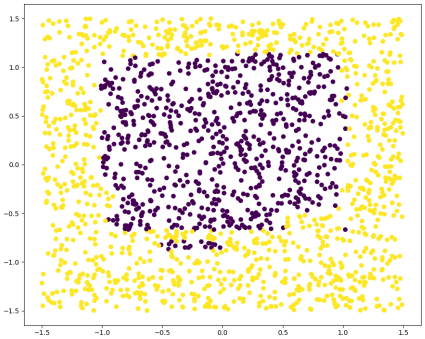
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from the candidate training set. The subscript n in D_n denotes training set size. The easiest way is to take the first n items from the (same) permutation above. This sequence simulates the real world situation where you obtain more and more training data.
   - For each D_n above, train a decision tree. Measure its test set error err_n. Show three things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n. This is known as a learning curve (a single plot). (3) Visualize your decision trees' decision boundary (five plots).

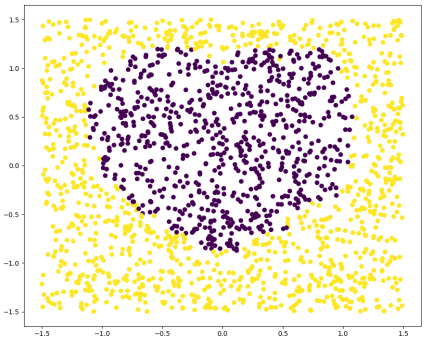
————————[D32, D128, D512, D2048, D8192]
Number of nodes: [11, 29, 85, 189, 329]
Number of Error: [298, 154, 124, 78, 29]

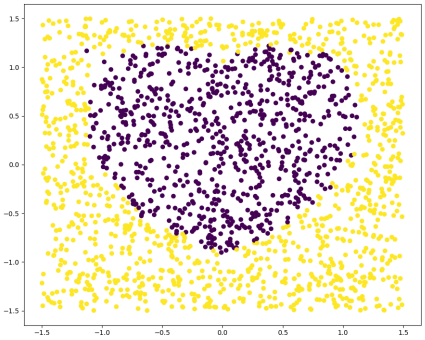






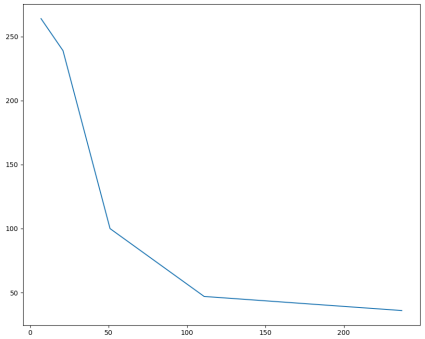






3 sklearn [10 pts]

Learn to use sklearn (https://scikit-learn.org/stable/). Use sklearn.tree.DecisionTreeClassifier to produce trees for datasets $D_{32}, D_{128}, D_{512}, D_{2048}, D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n.
numNodes = [7, 21, 51, 111, 237]
numErrors = [264, 239, 100, 47, 36]



4 Lagrange Interpolation [10 pts]

Fix some interval $[a, b]$ and sample $n = 100$ points x from this interval uniformly. Use these to build a training set consisting of n pairs (x, y) by setting function $y = sin(x)$.

Build a model f by using Lagrange interpolation, check more details in https://en.wikipedia.org/wiki/Lagrange_polynomial and https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.lagrange.html.

Generate a test set using the same distribution as your test set. Compute and report the resulting model's train and test error. What do you observe? Repeat the experiment with zero-mean Gaussian noise ϵ added to x. Vary the standard deviation for ϵ and report your findings.
No Noise, Training Error Rate: 5.941272521994731e+150
No Noise, Test Error Rate: 4.003531760948213e+34
Gaussian Noise with Var=1, Training Error Rate: 5.941272521994731e+150
Gaussian Noise with Var=1, Test Error Rate: 4.003531760948213e+34
Gaussian Noise with Var=2, Training Error Rate: 5.941272521994731e+150
Gaussian Noise with Var=2, Test Error Rate: 4.003531760948213e+34