HOMEWORK 2

>>Abhay Kumar<<
>>9081403157<<

Instructions: Although this is a programming homework, you only need to hand in a pdf answer file. There is no need to submit the latex source or any code. You can choose any programming language, as long as you implement the algorithm from scratch (e.g. do not use Weka on questions 1 to 7).
Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. 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:

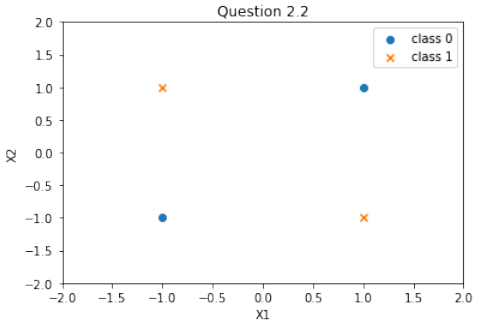
- Candidate splits (j, c) for numeric features should use a threshold c in feature dimension j in the form of $x_{\cdot j} \geq c$.
- c should be on values of that dimension present in the training data; i.e. the threshold is on training points, not in between training points.
- The left branch of such a split is the "then" branch, and the right branch is "else".
- Splits should be chosen using mutual information (i.e. information gain). 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 mutual information
- 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.

   Let's assume there are m items at the node with the same label (say y = 1).
   Entropy at the node = 0 as $p(y = 1) = \frac{m}{m} = 1$ and $p(y = 0) = 0$).
   Suppose we get a random split with the two halves having m_1 and $m - m_1$ items, so mutual information is given by $\frac{m_1}{m}(-\frac{m_1}{m_1}\log(\frac{m_1}{m_1}) - 0) + \frac{m-m_1}{m}(-\frac{m-m_1}{m-m_1}\log(\frac{m-m_1}{m-m_1}) - 0)$ = 0
   Hence, even if we split the node further, we won't get any information gain and thereby it is guaranteed to become a leave node.

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.



why ?
consider the following splits at root node and the information gain is zero in all the splits.
$X1 >= -1.000$ Information Gain=0.00000
$X1 >= 1.000$ Information Gain=0.00000
$X2 >= -1.000$ Information Gain=0.00000
$X2 >= 1.000$ Information Gain=0.00000

Since. there is no information gain, the decision tree algorithm will not split and make root as leaf and stop. The XOR function is inseparable in X1-X2 space as no horizontal or vertical split can divide the data into two halves with extra information gain. And, if we force a split, it will keep splitting till we get all the four points as leaf nodes and produces zero training error (overfitting to training data).

3. (Mutual information exercise) [10 pts] Use the training set Druns.txt. For the root node, list all candidate cuts and their mutual information. Hint: to get $\log_2(x)$ when your programming language may be using a different base, use `log(x)/log(2)`.

$X1 >= 0.100$ Information Gain=0.04418
$X1 >= 0.000$ Information Gain=0.00000
$X2 >= -2.000$ Information Gain=0.00000
$X2 >= -1.000$ Information Gain=0.04418
$X2 >= 0.000$ Information Gain=0.03827
$X2 >= 1.000$ Information Gain=0.00489
$X2 >= 2.000$ Information Gain=0.00108
$X2 >= 3.000$ Information Gain=0.01631
$X2 >= 4.000$ Information Gain=0.04945
$X2 >= 5.000$ Information Gain=0.10520
$X2 >= 6.000$ Information Gain=0.19959
$X2 >= 7.000$ Information Gain=0.03827
$X2 >= 8.000$ Information Gain=0.18905

The algorithm will not calculate Information gain on $X1 = 1, 2, 3, 4$ as these adjacent pairs in sorted partitions have instances with same class labels.
$X2 >= 6.000$ Information Gain=0.19959 - this split corresponds to the maximum information gain

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.

$[X_1 >= 10.000]$

$then : [label = 1]$
$else [X_2 >= 3.000]$

$then : [label = 1]$
$else : [label = 0]$

convert your tree to a set of logic rules.

if $([X_1 >= 10])$ **or** $([X_1 < 10]$ and $[X_2 >= 3]) \implies$ label = 1
if $([X_1 < 10])$ and $([X_2 < 3] \implies$ label = 0

$$y = \begin{cases} 1, & \text{if } ([X_1 >= 10]) \vee (\neg([X_1 >= 10]) \wedge [X_2 >= 3]) \\ 0, & \text{otherwise i.e. } ([X_1 < 10] \wedge [X_2 < 3]) \end{cases}$$

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 Weka style plaintext tree; 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.

     $[X2 >= 0.201829]$

     $then : [label = 1]$
     $else : [label = 0]$

   - 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.

     for $[X2 >= 0.201829]$ all points have label 1 and otherwise the label is 0. This means that the decision boundary is just a horizontal straight line passing at x2 = 0.201829. The horizontal line at x2 = 0.201829 splits into two halfspaces, with upper halfspace having label 1 and lower halfspace having label 0.

   - Build a decision tree on D2.txt. Show it to us.

     Format of tree (shown below):
     same indentation under a tree branch:
     first block = "then" branch
     second block = "else" branch

[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.

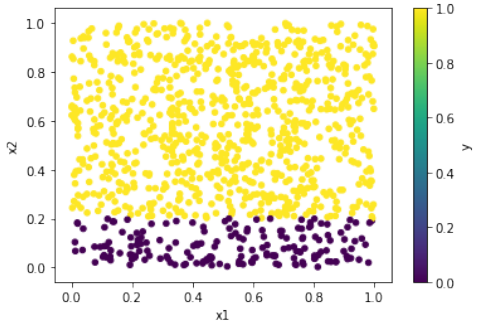
```
[X1 >= 0.533076]
   [X2 >= 0.383738]
      [X1 >= 0.550364]
         [label = 1.0]
         [X2 >= 0.474971]
            [label = 1.0]
            [label = 0.0]
      [X1 >= 0.761423]
         [X2 >= 0.191206]
            [label = 1.0]
            [X1 >= 0.904820]
               [X2 >= 0.037708]
                  [X1 >= 0.930371]
                     [label = 1.0]
                     [X1 >= 0.927522]
                        [label = 0.0]
                        [label = 1.0]
                  [label = 0.0]
               [X2 >= 0.169053]
                  [X1 >= 0.850316]
                     [label = 1.0]
                     [label = 0.0]
                  [label = 0.0]
         [X2 >= 0.301105]
            [X1 >= 0.663370]
               [label = 1.0]
               [label = 0.0]
            [label = 0.0]
   [X2 >= 0.639018]
      [X1 >= 0.111076]
         [X2 >= 0.861000]
            [label = 1.0]
            [X1 >= 0.330460]
               [label = 1.0]
               [X2 >= 0.745406]
                  [X1 >= 0.254049]
                     [label = 1.0]
                     [X1 >= 0.191915]
                        [label = 0.0]
                        [label = 0.0]
                  [label = 0.0]
         [X2 >= 0.964767]
            [label = 0.0]
            [label = 0.0]
      [X2 >= 0.534979]
         [X1 >= 0.409972]
            [X1 >= 0.426073]
               [label = 1.0]
               [label = 1.0]
            [label = 0.0]
         [label = 0.0]
```

- Try to interpret your D2 decision tree.

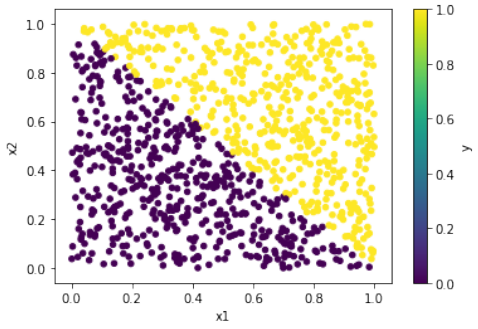
  It's very hard to interpret completely, but it seems that decision boundary passes through $X1 = 0.533$ and I noticed that the splits are mostly alternate between $X1$ and $X2$, showing that both splits are very similar in nature (This may be because of symmetric decision boundary w.r.t to some point). And the split values add up to one, i.e., if I consider $X1 >= 0.55036$ split, it follows with $X2 >= 0.47497$ split, and both splits values add to one. It gives hint that decision boundary may be just collection of points $(X1, 1 - X1)$. Additionally, there are too many horizontal and vertical splits making the decision boundary complex and hard to visualize completely.

6. (Hypothesis space) [10 pts] For D1.txt and D2.txt, do the following separately:

   - Produce a scatter plot of the data set.

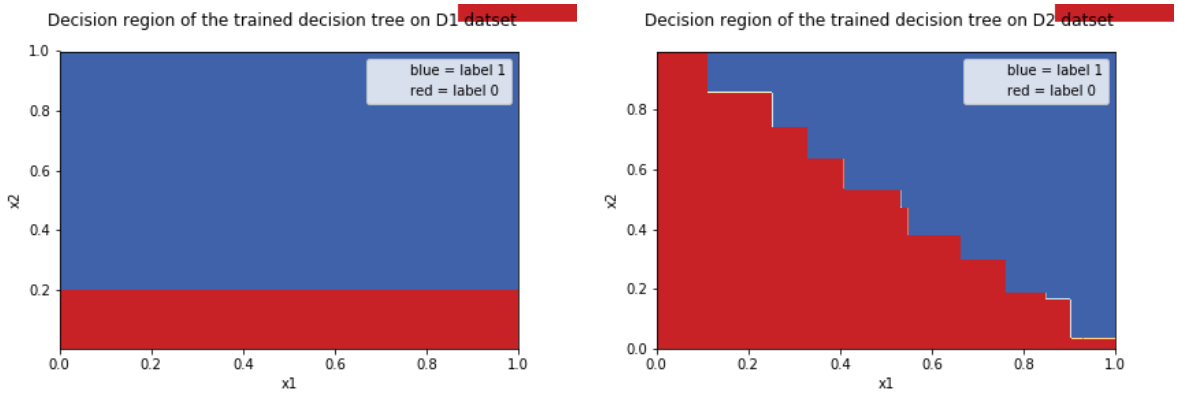


plot of D1.txt



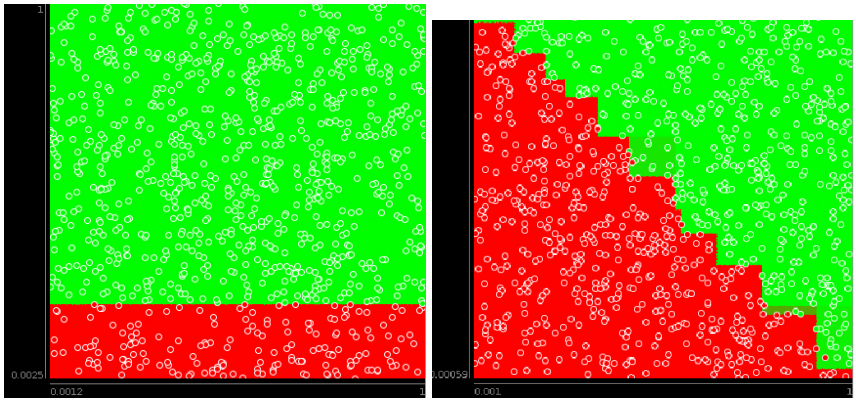
plot of D2.txt

- 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).



(a) (b)

Figure 5: Visualization of decision region on (a) D1 (b) D2 datasets respectively.



(a) (b)

Figure 6: Visualization of WEKA J48 decision region on (a) D1 (b) D2 datasets respectively.

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.

Hypothesis space of our decision tree algorithm needed to classify D1 data is -
$\theta_{D1} = (c) \in \mathcal{R}$ i.e., axis-parallel decision boundaries, i.e. either horizontal or vertical splits($x_1 >= c$ or $x_2 >= c$).

Hypothesis space of our decision tree algorithm needed to classify D2 data is -
$\theta_{D2} = (a, b, c) \in \mathcal{R}^3$ i.e., linear decision boundary ($ax_1 + bx_2 + c >= 0$). Since, this is more generic linear decision boundary and its representation needs combination of multiple horizontal and vertical splits, so the size of decision tree on D2 is more.

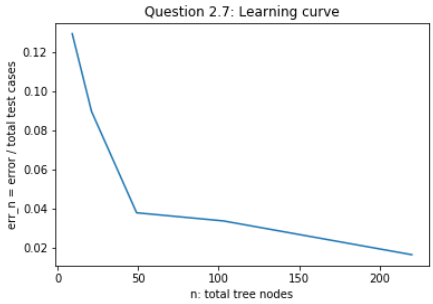
$\theta_{D1} \subset \theta_{D2}$: As θ_{D2} has additional parameters to be tuned for during training and therefore D2 Decision tree algorithm is bigger than that of D1.

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

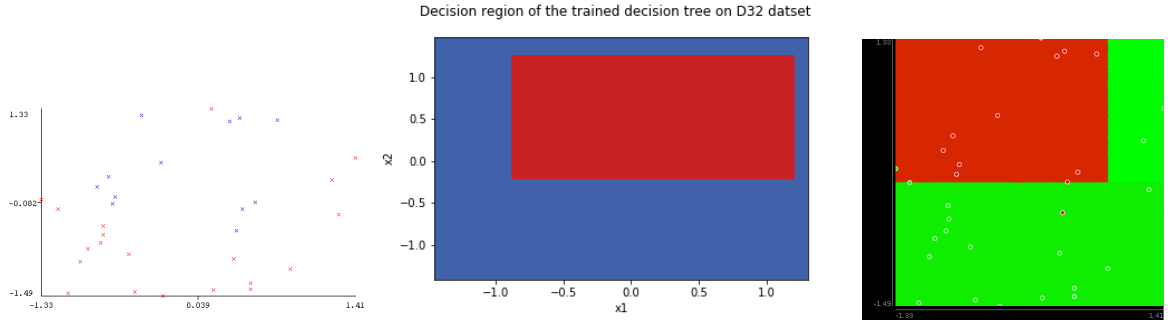
   - You will randomly split Dbig.txt into a candidate training set of 8192 items and a test set (the rest). Do this by generating a random permutation, and split at 8192.
   - Generate a sequence of five nested training sets $D_{32} \subset D_{128} \subset D_{512} \subset D_{2048} \subset D_{8192}$ from 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).

   considering $err_n = \frac{\text{error count}}{\text{Total test set size (=1908)}}$

D	n	err_n
32	9	0.1294
128	21	0.0896
512	49	0.0377
2048	103	0.0335
8192	220	0.0162



Plot n vs. err_n

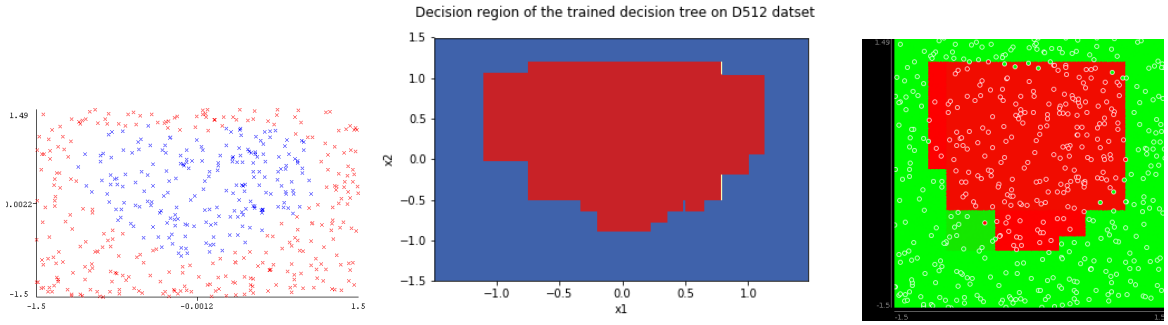


(a) (b) (c)

Figure 8: Visualization of (a) D32 (b) Decision boundary (c) WEKA decision boundary.

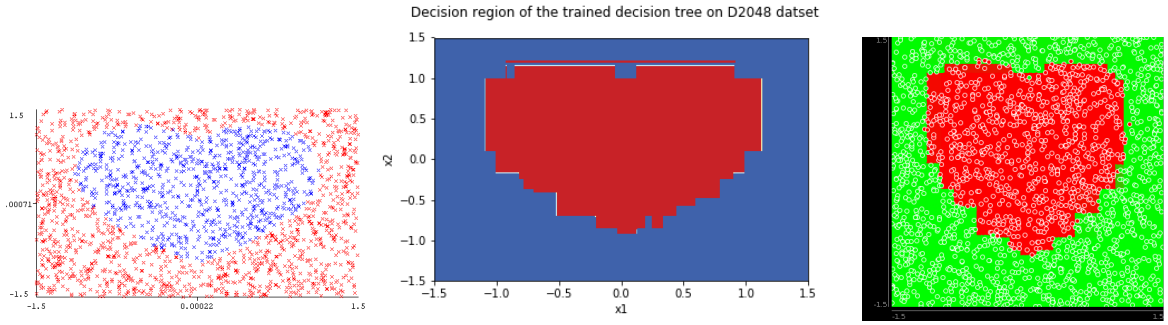
(a) (b) (c)

Figure 9: Visualization of (a) D128 (b) Decision boundary (c) WEKA decision boundary.



(a) (b) (c)

Figure 10: Visualization of (a) D512 (b) Decision boundary (c) WEKA decision boundary.



(a) (b) (c)

Figure 11: Visualization of (a) D2048 (b) Decision boundary (c) WEKA decision boundary.

(a) (b) (c)

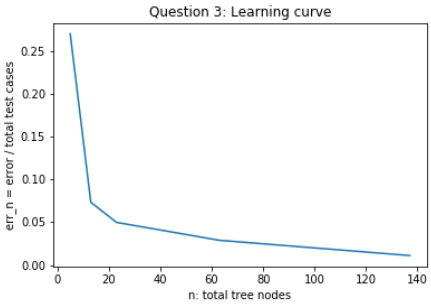
Figure 12: Visualization of (a) D8192 (b) Decision boundary (c) WEKA decision boundary.

3 Weka [10 pts]

Learn to use Weka https://www.cs.waikato.ac.nz/~ml/weka/index.html. Convert appropriate data files into ARFF format. Use trees/J48 as the classifier and default settings. Produce five Weka trees for $D_{32}, D_{128} \ldots D_{8192}$. Show two things in your answer: (1) List n, number of nodes in that tree, err_n. (2) Plot n vs. err_n.

considering $err_n = \frac{\text{error count}}{\text{Total test set size (=1908)}}$

D	n	err_n
32	5	0.2704
128	13	0.0733
512	23	0.0497
2048	63	0.0288
8192	137	0.0110



Plot n vs. err_n

WEKA trees visualization: Added in previous question (2.7)