CSE353 HW3

Yuqing

October 20, 2022

1. Machine learning fundamentals

   (a) The i.i.d. assumption is not reasonable. Those features are dependent and not identically. Since when a person have a fever presence, he is likely to get a headache. Those two symptoms are debilitating and make a person get sluggish behavior. Thus, those features cannot be assumed to be followed an independent and identically distribution.

   (b) The i.i.d. assumption is not reasonable. This measurement are not identically distributed across all the children but they are independent. It is because Johnny and Debbie are or have been suffering from diseases that may impair lung capacity. So the measurement between all of those children are not identical. But each child's lung capacity do not depend on others' lung capacity. Thus, the measurement is independent.

   (c) The i.i.d. assumption is not reasonable. Their chances of getting COVID are dependent and not identically. If their is a child getting COVID in the class of Amber's child, the probability for Amber and Brenda to get COVID will also increase and is different to Chloe's. Thus, their chances of getting COVID are not identically.

2. Entropy, conditional entropy, mutual information, information gain

2. Entropy, conditional entropy, mutual information, information gain (2 pts)

I have a very messy sock drawer, containing 10 red socks, 5 blue socks, 4 yellow socks, and 1 black sock.

(a) **(0.5 pts)** Recall the formula for entropy:

$$H(X) = -\sum_{X=x} \mathbf{Pr}(X = x) \log_2(\mathbf{Pr}(X = x)).$$

Define X a random variable which represents the color of a sock, randomly (uniformly) picked. What is th entropy of this sock?

$P(X=\text{red}) = \frac{1}{2}$
$P(X=\text{blue}) = \frac{1}{4}$
$P(X=\text{yellow}) = \frac{1}{5}$
$P(X=\text{black}) = \frac{1}{20}$

$H(X) = -\left(\Pr(X=\text{red}) \log_2 \left(P(X=\text{red}) \right)^{-1} \right.$
$+ \Pr(X=\text{blue}) \log_2 \left(P(X=\text{blue}) \right)^{-2}$
$+ \Pr(X=\text{yellow}) \log_2 \left(P(X=\text{yellow}) \right)^{-2.322}$
$\left. + \Pr(X=\text{black}) \log_2 \left(P(X=\text{black}) \right)^{-4.322} \right)$

$= 1.680$

(b) **(1 pt)** My mom comes and tells me I must organize my socks better. So, I put all my red socks in the top drawer and the rest in my bottom drawer. Recall the formula for conditional entropy:

$$H(X|Y) = -\sum_{X=x,Y=y} \mathbf{Pr}(X = x, Y = y) \log_2(\mathbf{Pr}(X = x|Y = y)).$$

What is the conditional entropy, where X is the color of a sock randomly picked, and Y is the drawer of which I pick it from? Assume that I pick the top drawer with twice the probability as picking the bottom drawer, but given a drawer, my choice of sock is uniformly distributed.

$P(Y = \text{top drawer}) = \frac{2}{3}$

$P(Y = \text{bottom drawer}) = \frac{1}{3}$

$H(X|Y) = -\big(P(\text{red}, \text{top}) \log_2 P(\text{red}|\text{top}) + P(\text{red}, \text{bot}) \log_2 P(\text{red}|\text{bot})$
$+ P(\text{blue}, \text{top}) \log_2 P(\text{blue}|\text{top}) + P(\text{blue}, \text{bot}) \log_2 P(\text{blue}|\text{bot})$
$+ P(\text{yellow}, \text{top}) \log_2 P(\text{yellow}|\text{top}) + P(\text{yellow}, \text{bot}) \log_2 P(\text{yellow}|\text{bot})$
$+ P(\text{black}, \text{top}) \log_2 P(\text{black}|\text{top}) + P(\text{black}, \text{bot}) \log_2 P(\text{black}|\text{bot}) \big)$

$= -\left(\frac{1}{2} \cdot \log_2 1 + 0 + 0 + \frac{1}{4} \times \log_2 \frac{1}{2} + 0 + \frac{1}{5} \times \log_2 \frac{2}{5} + 0 + \frac{1}{20} \cdot \log_2 \frac{1}{10} \right)$
$= -(-0.25 - 0.264 - 0.166)$
$= 0.68$

(c) $I(X;Y) = H(X) - H(X|Y) = 1.68 - 0.68 = 1$

3. Decision trees

(a) Computes entropy and conditional entropy for a sequence of labels.

entropy = 3.3141823231610834

conditional entropy = 3.3029598816135173

(b) Find best split

In this part, the best feature for the first split is feature 0 and split value for it is 2852.1111111111113.

The information gained in first step is 0.5969946869697194.

(c) My first split

printing tree...

0, ROOT, label 2.0, NONLEAF, split 0, val 2852.11

1, parent 0, label 2.0, LEAF, nsamples 119, purity 0.46

2, parent 0, label 1.0, LEAF, nsamples 349, purity 0.51

one step train err: 0.5

one step test err: 0.3138069912738163

(d) Pseudo-code for train error and test error for 25 steps of split.

// create tree by X_train and y_train

// X_train contains all the sample data for training.

// X_train[i] is a matrix for all the data for each feature in one sample data.

// y_train contains all the label for each data.

tree = Tree(X_train,y_train)

// loop for 25 steps for i in range(0,25):

// pop all the children of current node will be split in this iteration from tree.leaves to a new array leaves

leaves = []

for leaf in tree.leaves:

```
 leaves.append(tree.leaves.pop())
// Now, we get all the children of current node and the tree.leaves now are empty.
// Then, we loop all the leaf node in the leaves set.
for leaf_node in leaves:
    // check if the sample size of the leaf node is greater than 1
    if(len(leaf_node.sample_idx)>1):
        // sample size of leaf node is greater than 1
        // Find the best feature and split value for splitting
        // set1 will contain all the index of sample data in x whose value is smaller than or equal to the split value
        // set2 is a set contains all the index of sample data in x whose value is greater than the split value
best_feat, splitval, set1, set2 = find_best_split(X_train[leaf_node.sample_idx], y_train[leaf_node.sample_idx])

        // construct left and right child node by ids in set1 and set2
        left_child = tree.construct_node(set1)
right_child = tree.construct_node(set2)
        // Now, since we splitted the current leaf node, this node became a parent node for two child nodes. Thus, we set the is_leaf attribute for this node to False.
        leaf_node.is_leaf = False
        leaf_node.add_split_details(splitfeat = best_feat, splitval = splitval)

        // Then, extend tree.leaves set with left_child and right_child we get.
        leaf_node.children = [left_child, right_child]
        tree.leaves.extend(leaf_node.children)
    else:
        // The sample size of this leaf node is 1. It indicates that this node cannot be splitted anymore. Then we store this leaf node back to the tree.leaves
```

```
            tree.leaves.extend(leaf_node)
    // Print train error and test error for each step
    print('Step ', i+1, ' train err: ', tree.report_train_err())
    print('Step ', i+1, ' test err: ', get_test_err(tree))
    i += 1
// print the tree
tree.print_tree()
```

(e) Resulting Tree

|— feature$_0 \leq 2746.00$

| |$---feature_4 \leq 73.50$

| | |$---feature_0 \leq 2458.50$

| | | |$---feature_5 \leq 735.50$

| | | | |$---feature_5 \leq 243.50$

| | | | | |$---value : [2.33]$

| | | | |$---feature_5 > 243.50$

| | | | | |$---value : [3.14]$

| | | |$---feature_5 > 735.50$

| | | | |$---feature_6 \leq 225.00$

| | |$---value : [4.91]$

| | | | |$---feature_6 > 225.00$

| | | | | |$---value : [3.00]$

| | |$---feature_0 > 2458.50$

| | | |$---feature_2 6 \leq 0.50$

| | | | |$---feature_6 \leq 246.50$

| | | | | |$---value : [2.15]$

| | | | |$---feature_6 > 246.50$

| | | | | |$---value : [4.00]$

| | | |$---feature_2 6 > 0.50$

| | | | |$---feature_4 \leq 33.50$

| | | | | |$---value : [6.00]$

| | | | |$---feature_4 > 33.50$

| | | | | |$---value : [2.00]$

| |$---feature_4 > 73.50$

| | |$---feature_6 \leq 199.50$

| | | |$---feature_7 \leq 241.00$

| | | | | |--- $feature_4 6 \leq 0.50$

| | | | | | |--- $value$: [2.89]

| | | | | |--- $feature_4 6 > 0.50$

| | | | | | |--- $value$: [6.00]

| | | |--- $feature_7 > 241.00$

| | | | | |--- $value$: [6.00]

| | |--- $feature_6 > 199.50$

| | | |--- $feature_7 \leq 183.00$

| | | | | |--- $value$: [3.00]

| | | |--- $feature_7 > 183.00$

| | | | | |--- $feature_0 \leq 2734.50$

| | | | | | |--- $value$: [6.00]

| | | | | |--- $feature_0 > 2734.50$

| | | | | | |--- $value$: [5.00]

|--- $feature_0 > 2746.00$

| |--- $feature_0 \leq 3344.50$

| | |--- $feature_5 1 \leq 0.50$

| | | |--- $feature_0 \leq 2987.00$

| | | | | |--- $feature_3 6 \leq 0.50$

| | | | | | |--- $value$: [1.91]

| | | | | |--- $feature_3 6 > 0.50$

| | | | | | |--- $value$: [2.60]

| | | |--- $feature_0 > 2987.00$

| | | | | |--- $feature_4 8 \leq 0.50$

| | | | | | |--- $value$: [1.49]

| | | | | |--- $feature_4 8 > 0.50$

| | | | | | |--- $value$: [4.00]

| | |--- $feature_5 1 > 0.50$

| | | | $---feature_2 \leq 10.50$

| | | | | $---value : [1.00]$

| | | | $---feature_2 > 10.50$

| | | | | $---value : [7.00]$

| | $---feature_0 > 3344.50$

| | | $---feature_12 \leq 0.50$

| | | | $---feature_1 \leq 0.50$

| | | | | $---value : [7.00]$

| | | | $---feature_1 > 0.50$

| | | | | $---feature_0 \leq 3351.50$

| | | | | | $---value : [1.67]$

| | | | | $---feature_0 > 3351.50$

| | | | | | $---value : [1.00]$

| | | $---feature_12 > 0.50$

| | | | $---feature_7 \leq 248.00$

| | | | $|v|---value : [7.00]$

| | | | $---feature_7 > 248.00$

| | | | | $---value : [1.00]$

(f) Overfit

My tree overfits. In the figure 1, it shows 24 steps of split with the max information gain. The decision tree has a depth of 5. When looking through all the leaf nodes in Figure 2, we can find there are 25 leaf nodes and 5 of them have a sample size of 1. Besides, there are also many leaf nodes have small sample. Those are the hints to determine whether my decision tree is overfitting.

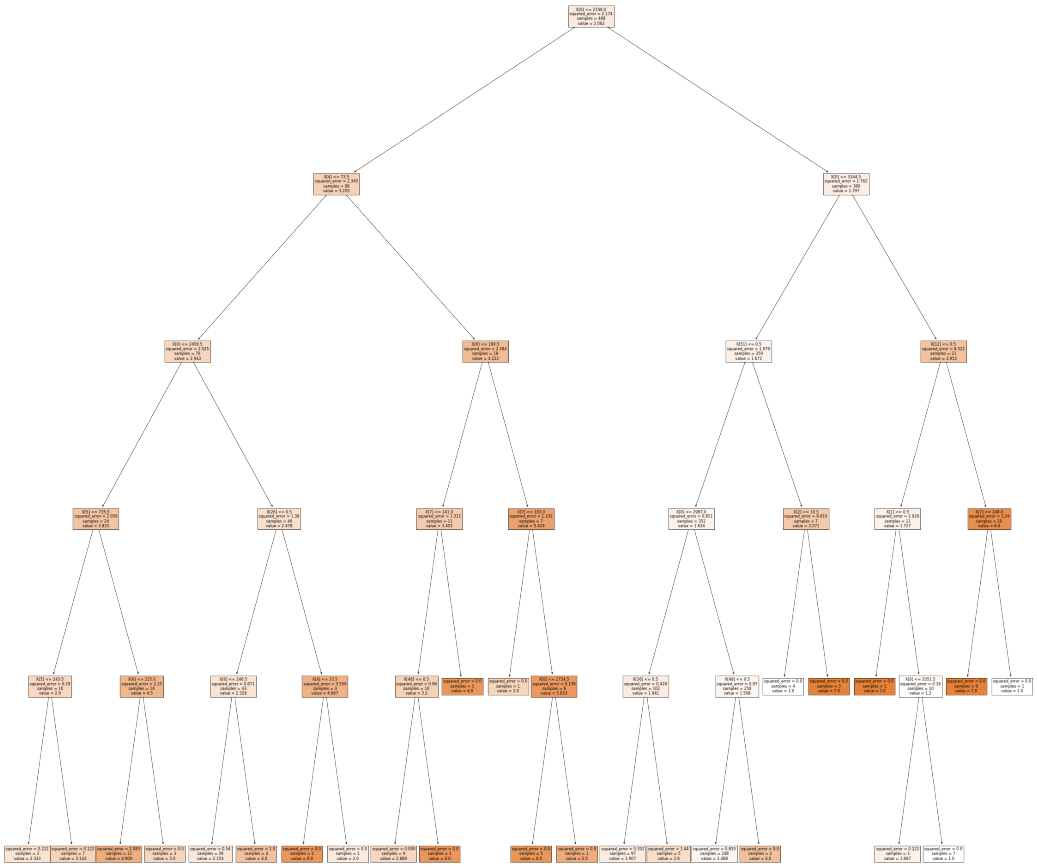

Figure 1: Decision Tree

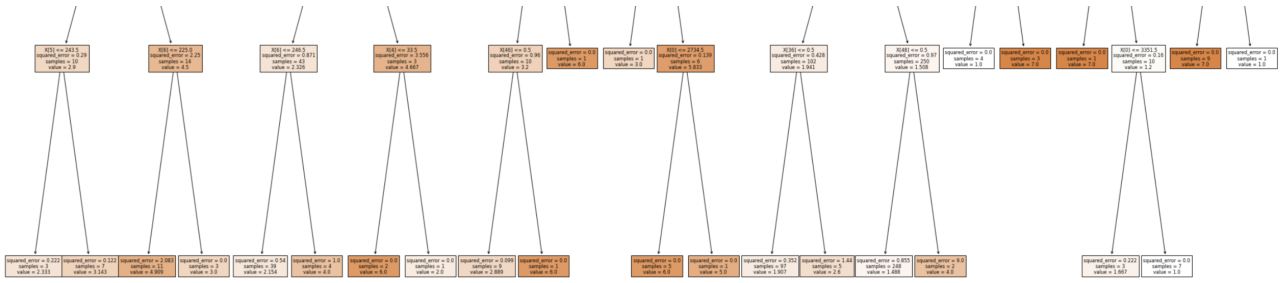

Figure 2: Leaves for My Decision Tree

4. Hidden Markov Model spellchecker

(a) Word probability

prob. of ”alice” 0.014548615047424706

prob. of ”queen” 0.002569625514869818

prob. of ”chapter” 0.0009069266523069947

(b) Transmission Matrix

prob. of ”the alice” 0.0

prob. of ”the queen” 0.03970678069639585

prob. of ”the chapter” 0.0

prob. of ”the hatter” 0.031154551007941355

(c) Find the 10 closest words to ’abice’

[’alice’, ’abide’, ’voice’, ’above’, ’alive’, ’twice’, ’dunce’, ’prize’, ’smile’, ’since’]

(d) Correction and recovery rate

According to the output above, the ratio of corrupt corpus before our correction to the correct corpus which is 0.783. 0.882 is the recovery rate of our ’fixed’ corpus which is the ratio of corrupt corpus after our correction to the correct corpus. It is obvious that our correction algorithm successfully reduce the misspelling.

Below those two rates, there are three columns of words. The first column shows the correct words. The second column lists the corrupt words correspond to the correct words. The third column lists the ’fixed’ words. So, words ’wonder’, ’pictures’, ’when’, ’sort’, ’little’, ’up’, ’take’, ’was’ are correctly fixed. But words ’by’, ’her’, ’mad’, ’didnt’, ’say’, ’feet’ are not correctly fixed.

```
0.783 0.882
wonder wondew wonder
by yb ye
by yb ye
her reh red
pictures pictuqes pictures
when whrn when
sort zort sort
mad mda ada
didnt ddint paint
say asy any
feet teef feel
little lvttle little
up uw up
take takd take
was wau was
```

Figure 3: Corpus Fixing output