CS251 HW 10 | Mon May 6, 2019 | Week 14

Name: Ethan Seal

Question 1: Decision Trees

a) Write a function to compute the entropy of an array of class counts. Test it using arrays that we would expect to give us the two extreme entropy values.

Implementation detail: You'll want to set a small non-zero minimum threshold on your probabilities otherwise you'll run into numerical stability issues when taking the logarithm.

```
import scipy.stats
def entropy(c):
    prob = np.maximum(c/np.sum(c), 0.001)
    return np.sum(-prob * np.log2(prob))
```

b) Follow the ID3 algorithm to construct a **1R** decision tree by hand using the below movie viewing data. The output classification classes are in the `Will watch` column (i.e. will the customer watch movie i). Use your entropy function to help you with the calculations.

Movie	Genre	Longer than 1 hour	Twitter account mentions	Age	Will watch
1	Comedy	Yes	10 <= x < 20	Old	Yes
2	Comedy	Yes	>20	Old	Yes
3	Comedy	Yes	<10	New	No
4	Comedy	No	>20	New	Yes
5	Drama	Yes	<10	New	No
6	Drama	No	10 <= x < 20	New	No
7	Drama	No	<10	Old	No
8	Sci-fi	Yes	>20	Old	Yes
9	Sci-fi	Yes	<10	New	No
10	Sci-fi	No	10 <= x < 20	New	Yes

Genre (5,5)

(3,1) Comedy | Drama (0,3) | Sci-Fi (2,1)

$1-(0.4(0.56)+0+0.3(0.63))=0.587$

Longer than 1 hr.

(3,3) yes | no (2,2)

$1-(\frac{1}{2}(1)+\frac{1}{2}(1))=0$

Twitter

(0,4) < 10 | (2,1) 10-20 | (3,0) > 20

$1-(0+0.3(0.63)+0)=0.811$

Age (5,5)

(2,4) New | old (3,1)

$1-(0.6(0.63)+0.4(0.56))=0.398$

c) Use your 1R tree to classify the following test data.

Movie	Genre	Longer than 1 hour	Twitter account mentions	Age	Will watch
1	Drama	No	10 <= x < 20 yes	New	No
2	Sci-fi	Yes	10 <= x < 20 yes	New	No
3	Comedy	Yes	<10 No	New	No
4	Drama	No	>20 yes	Old	Yes
5	Drama	No	<10 No	Old	Yes

d) What is the **training error**, defined as the proportion of cases misclassified in the *training set*?

10%.

e) What is the **generalization error**, defined as the proportion of cases misclassified in the *test set*?

60%.

Question 2: Neural Networks

a) Download the code for ADALINE from the CS251 website. Change the network to implement the Perceptron learning rule.

For comparison purposes, print out the L^2 norm (`np.linalg.norm`) of the raw error between weighted input sum with the true classes (not the misclassification rate). Test the updated network on the AND (should work) and XOR (should not work) inputs.

Duplicate the line "error = classes - y"
Add:
y[y<0] = -1
y[y>=0] = 1

b) Can the Perceptron learn OR inputs?

No

c) How does the choice of learning rule seem to affect convergence of each model (i.e. number of iterations you run the network for the error to stabilize)?

Perceptron is faster but less accurate

d) Can the Perceptron learn NAND inputs (not AND)? Are any differences that you found for OR different with NAND? yes, it can learn it. Not sure what the second question is asking

e) Make sure that you understand the difference between ADALINE (Widrow-Hoff) and Perceptron learning rules.

OK

7)a.

```
import scipy.stats
def entropy(class_counts):
    Probs=[]
    for i in range(Max(class_counts)):
        Prob = 0
        for j in range(len(class_counts)):
            if class_counts[j] == i:
                Prob += 1
        Prob = Prob/len(class_counts)
        Probs.append(prob)
        Probs.append(1-Prob)
    return stats.entropy(Probs)
```