RussEECS 649: PROBLEM SET #10

Reading:

- R&N Chapter 19.1-3, 19.6.1, 19.8-9
  [You can do PS10 with this reading and the Learning from Examples 1 & 2 lectures]
  [This seems a little longer than the last few homeworks. Plan accordingly.]

Total Points: 100

Notes:

- Submitted electronically (via Gradescope)
- This includes some programming.

Problem 10.1 [10 points] *Decision Trees for Logical Functions*
Give a decision tree to represent each of the following boolean functions:

a. A and (not B)
b. A or (B and C)
c. A xor B
d. (A and B) or (C and D)

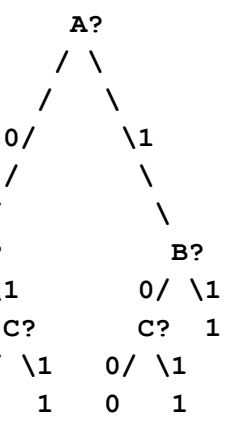
"xor" is 1 when either one, but not both, of its two arguments is 1.

Examples Like Problem 10.1. Here are a couple examples:

```
A or ((not B) or (C and (not D)))

          A?
        0/ \1
        B?  1
      0/ \1
      1   C?
        0/ \1
        0   D?
          0/ \1
          1   0
```

A	B	C	majority(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Problem 10.2 [10 points] *Entropy and Information Gain*
Consider this set of training examples:

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

Thus, instances 1, 2, and 4 are positive; 3, 5, and 6 are negative; a1 and a2 are the attributes

a. What is the entropy of this set of training examples with respect to the target function classification?
b. What is the information gain of a2 relative to these training examples?

Note: The "target function classification" is given by the "Classification" column; each row is an instance or item of training data.

Problem 10.3 [20 points] *Decision Tree Algorithms*

This is a question about the Decision-Tree-Learning Algorithm (aka ID3).

a. Show the decision tree that would be learned by ID3 assuming it is given the four training examples below for the EnjoySport? target concept.

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Positive and negative training examples for the target concept EnjoySport

b. Add the following training example, and compute the new decision tree. This time, show the value of the information gain for each candidate attribute at each step in growing the tree.

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
5	Sunny	Warm	Normal	Weak	Warm	Same	No

Note: For ease of grading and producing solutions, please break ties (among attributes with equal information gain) by choosing the attribute that appears first (in left to right order).

Problem 10.4 [5 points]

Consider an ensemble learning algorithm that uses simple majority voting among K learned hypotheses. Suppose that each hypothesis has error ε and that the errors made by each hypothesis are independent of the others'. Calculate a formula for the error of the ensemble algorithm in terms of K and ε, and evaluate it for the cases where K =5 and ε= 0.1.

Hints:

- See the text at the top of page 697 of R&N, 4e
- You are asked to compute the probability of the majority making an error. This happens when they all make an error, when 4 out 5 make an error, or when 3 of 5 make an error.

Example Like Problem 10.4

Suppose we have the majority voting among three learned hypotheses (K=3 using the parlance of the problem).

Assuming e is probability of any **one** of them making an error, and all three of them are independent, the probability of **majority** of the three making an error is

$e^3 + 3\ e^2(1-e)$

The term on the left is the probability that all three make an error simultaneously; the term on the right is the probability that two make an error and one does not. Note that the 3 comes from 3 = (3 choose 2).

Specifically, if I have experts A, B, and C, then we have the following complete sample space of joint outcomes:

```
W = wrong
R = right

P(A is W) = P(B is W) = P(B is W) = e
P(A is R) = P(B is R) = P(B is R) = (1-e)

  A B C  |  Majority is?  |  Probability
---------+----------------+----------------------
  W W W  |       W        |    e*e*e
  W W R  |       W        |    e*e*(1-e)
  W R W  |       W        |    e*(1-e)*e
  W R R  |       R        |    e*(1-e)*(1-e)
  R W W  |       W        |    (1-e)*e*e
  R W R  |       R        |    (1-e)*e*(1-e)
  R R W  |       R        |    (1-e)*(1-e)*e
  R R R  |       R        |    (1-e)*(1-e)*(1-e)
```

Problem 10.5 [15 points] *Function Fitting*
Find and **sketch** the following:

a. 1st-order (linear) Lagrange interpolating polynomial through (1,2) and (3,5)
b. 2nd-order (quadratic) Lagrange interpolating polynomial through (-1,0.1), (0,0), (1,0.1)
c. 1st-order (linear) least squares fit for points (-1,0.1), (0,0), (1,0.1)

Report the actual polynomials found, simplifying to a single number each of their coefficients.

Note: The formulas for Lagrange interpolation are at
https://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html.
The formulas for the least-squares fitting with a line are in Eq. (19.3) of R&N, 4e, and also at
http://mathworld.wolfram.com/LeastSquaresFitting.html.
In the second page, the form of the line is given in Equation (3) and the formulas for those constants are given in Equations (13) and (15).

The following problem requires some programming.

Problem 10.6 [40 points] *Programming Decision Stumps*
For this problem you will implement code to compute a "decision stump" (decision tree with just one test at the root; see definition in R&N, 4e, p. 700) for a real-life data set. Here is a linked example decision stump.

The data is a massaged version (to make all variables 0-1 valued and remove instances with missing values) of the Congressional Voting Records in the UCI Machine Learning database. There is the classification (Attribute 1; 0=Democrat, 1=Republican) and 16 attributes (Attributes 2-17), which details votes on various bills taken from the 1984 United States Congressional Voting Records (0=nay, 1=yes).

The massaged data you will use along with a more detailed description appear inside this linked Python notebook DecisionStumpBase.ipynb. The program also includes some useful items:

- Russell and Norvig's entropy function for boolean variables, *B* (see p. 662 of R&N, 4e)
- Code snippets for summing positive and negative examples of attributes

There is also a C++ version there, but it is not guaranteed and uses different functions as defined in R&N's Third Edition.

Perform the following

a. (10 points) Write a subroutine for *Remainder*(*A*), described on page 662 of R&N, 4e, that takes as its input value for A an integer between 2 and 17 corresponding to Attributes 2-17 as defined in the data file.
b. (10 points) Write a subroutine for *Gain*(*A*), again described on page 662 of R&N, 4e, that again takes an integer input as above.
c. (20 points) Write the necessary code to compute the decision stump. This entails (i) finding the attribute (2-17) that maximizes the information gain, (ii) counting how many of each classification appear in each branch, and (iii) computing the classifications you should use depending on that attribute's value. **So, you are splitting on the yeses and nays (for each vote) and trying to predict the Dems and Reps. IDEA: how can you best predict the rep.'s party by their vote on just one particular bill?**

We do **not** want to collect/handle/skim pages of common data and code. Thus, for parts (a) and (b), turn in **only** the code for each subroutine. (By the way, the use of global variables, like table is highly encouraged to ease programming.) For part (c), turn in **only** the code for your new/modified **main** program. (You may freely use or delete the code in main() of data.cc.)

Furthermore, for part (c), record the following in a **drawing**:

- the number of Democrat and Republican classifications in the whole tree,
- the attribute # and full name (from data file) used to split at the root,
- the number of Democrat and Republican classifications in each branch, and the
- prediction for each branch

NOTE: if you are trying to debug your code, you can experiment with the restaurant example in R&N. Here is a link to a CSV with the data so you don't have to retype it:

https://raw.githubusercontent.com/aimacode/aima-data/master/restaurant.csv