CARNEGIE MELLON UNIVERSITY

10601A/C-F18: Homework #2 - “Concept Learning”

TA: Swapnil Singhavi (ssinghav@andrew.cmu.edu)

Swapnil’s Office Hours:

Fri. 9/7	8pm-10pm	GHC 5 Commons near 5508
Sat. 9/8	3pm-5pm	GHC 5 Commons near 5508
Mon. 9/10	2pm-4pm	GHC 5 Commons near 5508
Tue. 9/11	2pm-4pm	GHC 5 Commons near 5508
Wed. 9/12	2pm-4pm	GHC 5 Commons near 5508
Thurs. 9/13	2pm-4pm	GHC 5 Commons near 5508

Assigned: Friday, 7 September 2018.
Due: 11:59:59pm on Thursday, 13 September 2018.
You are allowed up to 25 submissions.
Late Penalty: 20% per day.

Course Policies

Previously Used Assignments

Some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions, or elsewhere. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be, or may have been, available online, or from other people or sources. **It is explicitly forbidden to use any such sources, or to consult people who have solved these problems before. It is explicitly forbidden to search for these problems or their solutions on the internet.** You must solve the homework assignments completely on your own. **For programming assignments, this means you must write your programs completely by yourself, and not use any code from any source whatsoever.** I will be actively monitoring your compliance, and any violation will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

Collaboration Among Students

The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. **The actual solution must be done by each student alone.**

The purpose of programming assignments in this course is to make sure you truly understand the relevant techniques. In my more than 20 years of teaching, I have found no better way to achieve this than by having each student struggle by him/herself to implement these techniques âĂIJfrom scratchâĂİ. For this reason, in the case of programming assignments **all code must be written by each student alone.** We will strictly enforce this policy, by carefully inspecting your code using sophisticated detection techniques. You have been warned!

The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved. Specifically, each assignment solution must include answering the following questions:

- Did you receive any help whatsoever from anyone in solving this assignment? (Yes / No). If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4").
- Did you give any help whatsoever to anyone in solving this assignment? (Yes / No). If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2").
- Did you find or come across code that implements any part of this assignment? (Yes / No) (See below policy on "found code"). If you answered 'yes', give full details (book & page, URL & location within the page, etc.).

If you gave help after turning in your own assignment and/or after answering the questions above, you must update your answers before the assignment's deadline, if necessary by emailing the TA in charge of the assignment.

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism.

Policy Regarding "Found Code"

You are encouraged to read books and other instructional materials, both online and offline, to help you understand the concepts and algorithms taught in class. These materials may contain example code or pseudo code, which may help you better understand an algorithm or an implementation detail. However, when you implement your own solution to an assignment, you must put all materials aside, and write your code **completely on your own, starting "from scratch".** Specifically, you may not use any code you found or came across.**If you find or come across code that implements any part of your assignment, you must disclose this fact in your collaboration statement even if you didnâĂŹt use it.**

Duty to Protect One's Work

Students are responsible for pro-actively protecting their work from copying and misuse by other students. If a student's work is copied by another student, the original author is also considered to be at fault and in gross violation of the course policies. It does not matter whether the author allowed the work to be copied or was merely negligent in preventing it from being copied. When overlapping work is submitted by different students, **both students will be punished.**

To protect future students, do not post your solutions publicly, neither during the course nor afterwards.

Severe Punishment of Violations of Course Policies

All violations (even first one) of course policies will **always** be reported to the university authorities, will carry **severe** penalties, usually **failure** in the course, and can even lead to **dismissal** from the university. This is not an idle threat–it is my standard practice. You have been warned!

Contents

0 Introduction & General Instructions

This assignment consists of two parts. You will implement and run the Find-S and List-Then-Eliminate algorithms, and also answer some questions about the input and output of your programs.

The programs you write will be automatically graded using the CMU Autolab system. You may write your programs in **Python2**, **Python3**, **Java**, **C**, or **C++**. However, you should use the same language for all parts below. Note that if you are going to use Python3, you will need to submit a blank file called as 'python3ṫxt'

Depending on your choice of programming language, download from autolab the correct tar file ("Download handout"). The tar file will contain all the data that you will need in order to complete this assignment. In addition, you will also need to create the following files before you submit (see the sections below for more details):

- `partA.{py|java|c|cpp}`
- `partB.{py|java|c|cpp}`
- `partB5.txt`
- `collaboration.txt`
- `python3.txt` - **Only** if you intend to use **Python3**. **Not needed** for **Python2**

Make sure to answer the questions in the 'collaboration.txt' file as per the course collaboration policy.

Do not modify the structure of the directory or rename the files therein. Answer the questions below by completing the corresponding file(s), and then compress the five files into a tar 'hw2.tar' by running:

```
$ tar -cvf hw2.tar *.{py|java|c|cpp} *.txt
```

DO NOT put the above files in a folder and then tar that folder. You must compress the files directly into a tar file and submit it to the Autolab online.

You are allowed a maximum of 25 submissions until the deadline (see front page of this handout). **If you need an extension, please contact the TA-in-charge as soon as you are aware of such need and provide your reason.**

Besides this writeup, you are also provided with a handout tarball “hw2data.tar” containing all the data you need and some example output files. To untar the file, use command

```
$ tar -xvf hw2data.tar
```

Information about the example output files can be found in hw2data/README.txt. Please make sure to read this file before handing in your work to the Autolab.

In this assignment, you will work on the task of determining whether a person on the titanic survived or not based on a set of binary attributes.

For each part below, you are provided with a training dataset, including instances with different values for the attributes and a “Yes” or “No” label for each instance (“Yes” is the positive label). You are also provided with a development dataset to develop and test your code locally. **Note**: Your submitted code will be evaluated on the server using a different test dataset which you do not have access to. Your program should accept a file, the test data, as a command line argument. You can test your code by passing the development data file as an argument. The test data file has the same format as the development data file.

Using three different datasets is typical in machine learning tasks. Training data is used to inductively learn a hypothesis. Training often involves fitting many parameters, although in this assignment it involves filtering hypotheses. Development data is often used for initial testing, deciding among several alternatives, and possibly for tuning of a few additional parameters. In this assignment, you will use it primarily to test your the performance of your code. A third dataset, test data, is used to evaluate the system and report its performance and should not, therefore, be used in training or developing the system.

1 Part A: The 'Titanic-9Cat' task and the FIND-S algorithm

In this part, you are provided with a training dataset "9Cat-Train.labeled" including examples with different values for 9 binary attributes and a "Yes" or "No" label for each example. The task is to learn a hypothesis that best fits the data. For the Titanic-9Cat task, the attributes are:

- Age (Young, Old)
- Cabin (A, B)
- Class (1, 3)
- CrewMember? (Yes, No)
- Embarked (Southampton, Queenstown)
- Sex (Male, Female)
- Traveling with Sibling? (Yes, No)
- Traveling with Parent/Spouse? (Yes, No)
- Near RescueBoat? (Yes, No)

In this part, your hypothesis space *H* should be exactly like the first one we discussed in class. Namely, it should consist of all possible conjunctions over the nine attributes, where each conjunct is of the form **ATTR=[value]**, **ATTR="?"**, or **ATTR="null"**. In your program, you can choose to exclude the possibility **ATTR="null"**, and find an alternative way to represent the single hypothesis that always returns "False".

You are also provided with a development dataset "9Cat-Dev.labeled" to develop and test your code. The command which will be used to run your program on the server, depending on the programming lanaguage that you use, is:

For Python2:

```
$ python partA.py testFileName
```

For Python3:

```
$ python3 partA.py testFileName
```

For Java:

```
$ java partA.java testFileName
```

For C:

```
$ gcc partA.c; ./a.out testFileName
```

For C++:

```
$ g++ partA.cpp; ./a.out testFileName
```

In the file `partA.{py|java|c|cpp}`, write a program to do the following:

1. Print (to `stdout`), by itself on the first line, the size of the input space (number of possible unique inputs).

2. Let $|C|$ = the size of the concept space (the space of all possible concepts; i.e., Boolean functions of the input). Print, by itself on the next line, the number of decimal digits in $|C|$.

   For example, if the size of the concept space is 128, the value printed should be 3 which is the number of decimal digits in 128.

3. Print, by itself on the next line, the size of the hypothesis space H (the space of all semantically distinct conjunctions of the type described above).

4. Suppose we add a new binary feature to our dataset (e.g., "Know Swimming? (Yes, No)"). Print, by itself on the next line, the size of this new hypothesis space.

5. Suppose that, instead of adding a new feature, one of the existing features can take on three different values (e.g., "Class" could be "(1, 2, 3)"). Print, by itself on the next line, the size of this new hypothesis space.

6. Run the FIND-S algorithm on "9Cat-Train.labeled", using the original hypothesis space H described above, and print to the file "partA6.txt" the current hypothesis after every 20 training instances (namely, after 20, 40, 60,... training instances, counting both positive and negative instances). A hypothesis is displayed as a tab-delimited list of attribute values. The current hypothesis after every 20 instances should be printed on a separate line. Check the example output file 'exampleA6.txt' for the expected format and order of the values.

7. Apply the final hypothesis to "9Cat-Dev.labeled" then print a line with the misclassification rate (the fraction of misclassified data points; a number between 0.0 and 1.0). Note that you are provided with the correct labels in "9Cat-Dev.labeled" so you can compare the predicted labels with the correct labels and compute the misclassification rate.

8. Apply the final hypothesis to the data in the input file (passed as a command line argument), and print out the classification of each instance in this set on a separate line.

Note that tasks 1-5 are "hardcoded" tasks while the other three are computation-based (i.e. you need to write code). Check the example output file "exampleA.txt" for the expected format of the program output. Please pay attention to whitespaces.

2 Part B: Bias-free Learning, the ‘Titanic-4Cat’ task and the LIST-THEN-ELIMINATE algorithm

In this part, you will attempt to learn without any bias. Your hypotheses space H will be equal to the concept space. Namely, *H* contains not only conjunctions but also every other possible binary function. The task is to run the LIST-THEN-ELIMINATE algorithm on the training data, keeping track of the version space, and then, for every test instance, hold a vote among the members of the final version space.

To reduce the computational complexity, we reduce the task to involve only the first four attributes. Thus, for the ‘Titanic-4Cat’ task, the attributes are:

- Age (Young, Old)
- Class (1, 3)
- Embarked (Southampton, Queenstown)
- Sex (Male, Female)

You are provided with a training dataset *D*=“4Cat-Train.labeled” including instances with different values for the 4 attributes and a “Yes” or “No” label for each such instance. You are also provided with a development dataset “4Cat-Dev.labeled” to develop and test your code. Your program should take one command line argument which is a test data file that has the same format as the development data file. The command which will be used to run your program on the server, depending on the programming language that you use, is:

For Python2:

```
$ python partB.py testFileName
```

For Python3:

```
$ python3 partB.py testFileName
```

For Java:

```
$ java partB.java testFileName
```

For C:

```
$ gcc partB.c; ./a.out testFileName
```

For C++:

```
$ g++ partB.cpp; ./a.out testFileName
```

In the file `partB.{py|java|c|cpp}`, write a program to do the following:

1. Print (to `stdout`), by itself on the first line, the size of the input space (number of possible unique inputs).

2. Print, by itself on the next line, the size of the concept space (the space of all possible concepts; i.e., Boolean functions of the input).

3. Run the 'List-Then-Eliminate' algorithm using the entire concept space, on D='4Cat-Train.labeled', settle on a version space $VS(H, D)$, then print out, on a new line, the size of that version space (number of remaining hypotheses consistent with the training data).

4. For each instance in the test data (where the test file is passed as a command line argument), take a vote among the hypotheses in the version space, and print a line containing two numbers: #Yes #No, separated by a space (e.g., "5 2").

5. What did you observe, and how well did your program do? What, if anything, can you learn from this? Save your answer to this question (B5) in the file 'partB5.txt'.

Note that tasks 1 and 2 are "hardcoded" tasks. Check the example output file 'exampleB.txt' for the expected format of the program output. Please pay attention to whitespaces.

3 Some Logistical Information

Please ensure you have completed the following files for submission and follow the instructions described in the "General Instructions" section to submit:

```
partA.{py|java|c|cpp}
partB.{py|java|c|cpp}
partB5.txt
collaboration.txt
```

Note: Please make sure the programming language that you use is consistent within this assignment (e.g. don't use C++ for part A and Python for part B).

Task B5 will be manually graded after all submissions have been made. Before that point, it will appear as 0/2 points.