CARNEGIE MELLON UNIVERSITY

10601a: Homework #3 - “Concept Learning”

TAs-in-charge:
Udbhav Prasad (udbhavp@andrew)
Xiaote Zhu (xiaotez@andrew)

Assigned: Monday, 26 January 2015.
Due: 11:59:59pm on Sunday, 1 February 2015.
Late Penalty: 25% per day.

Office Hours: Gates 5th Commons

	Xiaote Zhu	Udbhav Prasad
Monday 1/26	3:30 - 4:30pm	9:30 - 10:30pm
Tuesday 1/27	4:30 - 5:30pm	9:30 - 10:30pm
Wednesday 1/28	3:30 - 4:30pm	9:30 - 10:30pm
Thursday 1/29	2 - 3pm	9:30 - 10:30pm
Friday 1/30	2:30 - 3:30pm	11:30am - 12:30pm
Saturday 1/31	1 - 2pm	-
Sunday 2/1	1 - 3pm	8:30 - 10pm

Policy on Collaboration among Students

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. 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. **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. 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**, and the student should be ready to reproduce their solution upon request. In the case of programming assignments, **all code must be written by each student alone**. We will strictly enforce this policy. **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 must contain a file named Collaboration.txt where you will answer 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 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 to section 2.3 to help him with Question 2").

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.

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**.

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!

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 Python or Java. However, you should use the same language for all parts below.

Depending on your choice of programming language, download from autolab the correct tar file “hw3Python.tar”, or “hw3Java.tar”. The tar file contains each of the files you need to complete this assignment in addition to ’Collaboration.txt’. 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, then compress the five files into a tar directory “hw3Java.tar” or ‘’hw3Python.tar” by running

For Python:

```
tar -cvf hw3Python.tar *.py *.txt
```

For Java:

```
tar -cvf  hw3Java.tar  *.java *.txt
```

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

You are allowed a maximum of 25 submissions until the deadline.

You are also provided with another tar file, “hw3data.tar”, containing data and example output files. Information about the example output files can be found in hw3data/README.txt. Please make sure to read this file before interpreting the contents of the provided example output.

0 Introduction

In this assignment, you will work on the task of determining whether the risk of defaulting on a loan is “high” or “low,” 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 “high” or “low” label for each instance (“high” is the positive label). You are also provided with a development dataset to develop and test your code. Your submitted code will be tested on the server using a different, test dataset that 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 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 "Risk-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 "high" or "low" label for each example. The task is to learn a hypothesis that best fits the data. For the Risk-9Cat task, the attributes are:

- Gender (Male, Female)
- Age (Young, Old)
- Student? (Yes, No)
- Previously Declined? (Yes, No)
- Hair Length (Long, Short)
- Employed? (Yes , No)
- Type Of Colateral (House, Car)
- First Loan (Yes, No)
- Life Insurance (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 is basically:

For Python:

```
python partA.py testFileName
```

For Java:

```
java partA testFileName
```

In the file "partA", write a program that:

1. Prints (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). Prints, 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. Prints, 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. Runs the FIND-S algorithm on “9Cat-Train.labeled”, using the hypothesis space H described above, and prints to the file "partA4.txt" the current hypothesis after every 30 training instances (namely, after 30, 60, 90,... training instances, counting both positive and negative instances). A hypothesis is displayed as a tab-delimited list of attribute values (Please represent NULL with the string “null”, keeping in mind that the autgrader is case-sensitive. The attribute values must be represented as strings as they are present in the input data. For example, one value of the attribute Gender is Male, not MALE or male). The current hypothesis after every 30 instances should be printed on a separate line. Check the example output file ’partA4.txt’ for the expected format

5. Applies the final hypothesis to “9Cat-Dev.labeled” then prints 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.

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

Check the example output file “exampleA.txt” for the expected format of the program output. Please pay attention to whitespaces and our autograder is case-sensitive.

2 Part B: Bias-free Learning, the ’Risk-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 ’Risk-4Cat’ task, the attributes are:

- Gender (Male, Female)

- Age (Young, Old)
- Student? (Yes, No)
- Previously Declined? (Yes, No)

You are provided with a training dataset D="4Cat-Train.labeled" including instances with different values for the 4 attributes and a "high" or "low" 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 have 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 is basically:

For Python:

```
python partB.py testFileName
```

For Java:

```
java partB testFileName
```

In the file "partB", write a program that:

1. Prints (to `stdout`), by itself on the first line, the size of the input space (number of possible unique inputs).
2. Prints, 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. Runs the 'List-Then-Eliminate' algorithm, using the entire concept space, on D='4Cat-Train.labeled', settles on a version space VS(H,D), then prints 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), takes a vote among the hypotheses in the version space, and prints a line containing two numbers: #high, #low, separated by a space.
5. How well did it do? What, if anything, can you learn from this? Save your answer to this question (B5) in the file 'partB5.txt'

Check the example output file 'exampleB.txt' for the expected format of the program output. Please pay attention to whitespaces and our autograder is case-sensitive.