CS 307 Introduction to Machine Learning
Fall 2022
Assignment 2: Bayesian Learning
Part I: *Due electronically by Monday, October 17, 11:59 p.m.*
Part II: *Due electronically by Monday, October 24, 11:59 p.m.*

Instructions:

1. Your solution to this assignment must be submitted via gradescope.
2. For the written problems, submit your solution to the *Assignment 2 - Part 1 (CS 307)* folder as a **single PDF** file.
   - We prefer that you type your solution. If you choose to hand-write your solution, scan your PDF using a **scanner** and upload it. Make sure your final PDF is **legible**. **Regrades due to non-compliance will receive a 30% score penalty**.
   - Verify that both your answers and procedure are **correct, ordered, clean, and self-explanatory** before writing. Please ask yourself the following questions before submitting:
     - Are my answers and procedure legible?
     - Are my answers and procedure in the same order as they were presented in the assignment? Do they follow the specified notation?
     - Are there any corrections or scratched out parts that reflect negatively on my work?
     - Can my work be easily understood by someone else? Did I properly define variables or functions that I am using? Can the different steps of my development of a problem be easily identified, followed, and understood by someone else? Are there any gaps in my development of the problem that need any sort of justification (be it calculations or a written explanation)? Is it clear how I arrived to each and every result in my procedure and final answers? Could someone describe my submission as messy?
3. **IMPORTANT:** As long as you follow these guidelines, your submission should be in good shape; if not, we reserve the right to penalize answers and/or submissions as we see fit.

Part I: Written Problems (70 points)

1. **Probability I (14 points)**

   Suppose that A and B are binary-valued random variables. We know that $P(A)$ = 0.7, $P(B)$ = 0.8, and $P(B|A)$ = 0.9.

   (a) What is $P(A|B)$?

   (b) What is $P(B| \sim A)$?

   Show and explain your work for both (a) and (b).

2. **Probability II (14 points)**

   Suppose you are given n (six-sided) dice. You are told that $n - 1$ of them are fair (i.e., the probability of getting each of the six outcomes is $\frac{1}{6}$), whereas the remaining one is *two-biased* (i.e., the probability of getting a '2' equal to 1).

   (a) Suppose you pick out a die uniformly at random, throw it, and get a '2'. What is the probability that you picked the two-biased die?

   (b) Suppose you continue throw the die for a total of k times after picking it and get k '2's. Now what is the probability that you picked the two-biased die?

3. **Bayesian Classifiers (20 points)**

   We want to learn the concept of `graduate student`. Consider the following set of training examples.

`Class`	`good hacker?`	`has publications`	`hobby`
positive	yes	yes	sci-fi
negative	yes	no	music
positive	no	yes	sci-fi
positive	yes	no	tennis
negative	no	yes	music
positive	no	no	tennis
positive	yes	yes	sci-fi
negative	yes	yes	tennis
positive	no	no	sci-fi
negative	no	yes	tennis

   (a) How would a naive Bayes classifier trained on this training set classify the following instances?

   (i) `hobby`=tennis $\wedge$ `good-hacker`=yes $\wedge$ `has-publication`=yes

   (ii) `hobby`=tennis $\wedge$ `good-hacker`=yes $\wedge$ `has-publication`=no

(b) How would a Bayes classifier trained on this training set classify the following instances?

(i) `hobby`=tennis $\wedge$ `good-hacker`=yes $\wedge$ `has-publication`=yes

(ii) `hobby`=tennis $\wedge$ `good-hacker`=yes $\wedge$ `has-publication`=no

4. **Hypothesis Selection (22 points)**

Consider a medical diagnosis problem in which there are two alternative hypotheses: (1) that the patient has a particular form of coronavirus, and (2) that the patient does not. The available data is from a particular laboratory test with two possible outcomes: *positive* and *negative*. We have prior knowledge that over the entire population of people only 0.007 have this disease. Furthermore, the lab test is only an imperfect indicator of the disease. The test returns a correct positive result in only 97% of the cases in which the disease is actually present and a correct negative result in only 96% of the cases in which the disease is not present. In other cases, the test returns the opposite result.

(a) Suppose we now observe a new patient for whom the lab test returns a positive result.

(i) Should we diagnose the patient as having coronavirus or not if we take into account our prior knowledge?

(ii) Should we diagnose the patient as having coronavirus or not if we **do not** take into account our prior knowledge (i.e., we assume that the priorprobability that someone in the entire population has the disease is 0.5)?

(b) Suppose the doctor decides to order a second lab test for the same patient, and suppose the second test returns a positive result as well. Assume that the two tests are independent.

(i) Should we diagnose the patient as having coronavirus or not if we take into account our prior knowledge?

(ii) Should we diagnose the patient as having coronavirus or not if we **do not** take into account our prior knowledge (i.e., we assume that the priorprobability that someone in the entire population has the disease is 0.5)?

Part II: Programming (30 points)

In this problem, you will implement the Naive Bayes learning algorithm for binary classification tasks (i.e. each instance will have a class value of 0 or 1). To simplify the implementation, you may assume that all attributes are binary-valued (i.e. the only possible attribute values are 0 and 1) and that there are no missing values in the training or test data.

A sample training file (`train.dat`) and test file (`test.dat`) are available on the course website on Canvas. In these files, only lines containing non-space characters are relevant. The first relevant line holds the attribute names. Each following relevant line defines a single example. Each column holds this example's value for the attribute named at the head of the column. The last column (labeled "class") holds the class label for the examples.

IMPORTANT:

- To implement the Naive Bayes classifier, you should use C++ (g++ 7.5.0), Java (openjdk 11.0.13 2021-10-19), or Python (3.6.9). Do **not** use any non-standard libraries (except numpy (1.19.5) and pandas (1.1.5)) in your code.
- Your program should be able to handle **any** binary classification task with **any** number of binary-valued attributes. Consequently, both the number and names of the attributes, as well as the number of training and test instances, should be determined at runtime. In other words, these values should **not** be hard-coded in your program.
- Your program should allow exactly two arguments to be specified in the command line invocation of your program: a training file and a test file. Your program should take these two arguments in the same order as they are listed above. There should be no graphical user interface (GUI). Any program that does not conform to the above specification will receive no credit.
- In the input files, only lines containing non-space characters are relevant, as mentioned previously. In particular, empty lines may appear anywhere in an input file, including the beginning and the end of the file. Care should be taken to skip over these empty lines.
- You may submit as many source files as needed, but you must make sure you provide a main code entry that follows the following naming convention. Specifically, if you are using:
  - **Python**
    - Make sure that your primary source file is `main.py` and that your code runs successfully after executing `python main.py <path_to_train_file> <path_to_test_file>`.
  - **C++**
    - Make sure that your primary source file is `main.cpp` and that your code runs successfully after executing `g++ main.cpp -o a.out -std=c++17` and `./a.out <path_to_train_file> <path_to_test_file>`.
  - **Java**
    - Make sure that your primary source file is `Main.java` and that your code runs successfully after executing `javac Main.java` and `java Main <path_to_train_file> <path_to_test_file>`.

What to Do

1. Train the Naive Bayes learning algorithm on the training instances. Print to `stdout` the parameters of the classifier that were estimated from the training data (i.e., the class priors and the class-conditional probabilities). To exemplify, assume that the class attribute is named C and has two possible values, c1 and c2; furthermore, assume that there are three attributes A1, A2, and A3, where A1 has two possible values (x1 and x2), A2 has two possible values (y1 and y2), and A3 has three possible values (z1, z2, and z3). The learned parameters should be printed according to the following format:

P(C=c1)=0.32 P(A1=x1|c1)=0.33 P(A1=x2|c1)=0.67 P(A2=y1|c1)=0.25 P(A2=y2|c1)=0.75 P(A3=z1|c1)=0.26 P(A3=z2|c1)=0.49 P(A3=z3|c1)=0.25
P(C=c2)=0.68 P(A1=x1|c2)=0.22 P(A1=x2|c2)=0.78 P(A2=y1|c2)=0.91 P(A2=y2|c2)=0.09 P(A3=z1|c2)=0.16 P(A3=z2|c2)=0.74 P(A3=z3|c2)=0.10

In other words, all the probabilities associated with a particular class should appear in the same line.

2. Use the learned classifier to classify the **training** instances. Print to `stdout` the accuracy of the classifier. The accuracy should be computed as the percentage of examples that were correctly classified. For example, if 86 of 90 examples are classified correctly, then the accuracy of the classifier would be 95.6%.

   ```
   Accuracy on training set (90 instances):  95.6%
   ```

3. Use the learned classifier to classify the **test** instances. Print to `stdout` the accuracy of the classifier.

   ```
   Accuracy on test set (10 instances):  60.0%
   ```

Submission

Once you are done, sign in to gradescope. You will be able to see *Assignment 2 - Part 2 (CS 307)* under the Assignments section. Directly submit all your source files to this submission folder. Do not create any folder and do not rename the files or upload the files in a zip file or folder (your homework will not be graded otherwise).

Grading

We will be using an output-based auto-grader for this submission, so make sure you follow the formatting from the example test files: be careful not to insert extra lines, tabs instead of spaces, etc ... When you submit, your code will be **graded using hidden test cases**, so we encourage you to test your code thoroughly. More information about the autograder will be available on Piazza shortly.