CS 540: Introduction to Artificial Intelligence

Homework Assignment #5

Assigned: Tuesday, April 11
Due: Tuesday, April 25

Hand-in Instructions

This assignment includes written problems and programming in Java. **The homework problems must be done individually.** Hand in all parts electronically by copying them to the Moodle Dropbox called "`HW #5 Hand-In`". Your answers to each written problem should be turned in as separate pdf files called `<wisc NetID>-HW5-P1.pdf`, `<wisc NetID>-HW5-P2.pdf and <wisc NetID>-HW5-P3-2.pdf` (If you write out your answer by hand, you'll have to scan your answer and convert the file to pdf). Put your name at the top of the first page in each pdf file.

For the programming problem, put all the java files needed to run your program, including ones you wrote, modified or were given and are unchanged (including the required `HW5.java`), into a folder called `<wisc NetID>-HW5.` Compress this folder to create `<wisc NetID>-HW5-P3.zip` and copy this file to the Moodle Dropbox. **Make sure your program compiles and runs without any error on CSL machines**. Your program will be tested using several test cases of different sizes. Make sure all files are submitted to the Moodle Dropbox.

Late Policy

All assignments are due **at 11:59 p.m.** on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday and it is handed in between any time on Thursday, 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline. A total of three (3) free late days may be used throughout the semester without penalty. Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Problem 1. [15] Conditional Independence and Joint Probability

One advantage of Bayesian networks is their ability to represent high-dimensional distributions compactly. We will examine this in this problem. Consider 4 random variables *A*, *B*, *C* and *D*. Each variable has 3 possible values. Answer these questions and briefly show your work.

(a) [3] *At least* how many values are needed to uniquely specify the joint probability table on these variables? In general, how many values are needed for *n* 3-value random variables?

(b) [3] Given the Bayesian network A → B → C → D, *at least* how many values are needed to be associated with this network in order to uniquely determine (implicitly) the joint probability table?

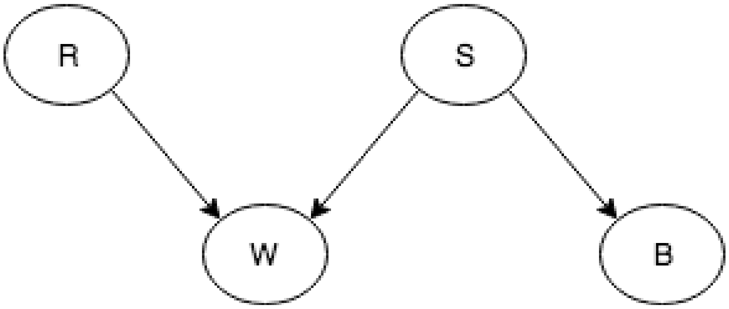
(c) [3] Using the same network given in (b), are *A* and *D* independent? Are *A* and *D* conditionally independent?

(d) [3] What is the *minimum* number of values needed to specify the joint probability table over *all* possible networks containing four Boolean random variables (assuming *any* independence and conditional independence assumptions)?

(e) [3] Draw the Bayesian network for (d).

Problem 2. [20] Bayesian Network Inference

Consider the following Bayesian Network containing four Boolean events (random variables): *R* = "it rained," *S* = "sprinkler on," *W* = "lawn wet," and *B* = "bird on lawn."



The associated CPTs for this network are defined as follows:

- $P(R) = 0.1$
- $P(S) = 0.4$
- $P(B \mid S) = 0.01$, $P(B \mid \neg S) = 0.5$
- $P(W \mid R, S) = 1$, $P(W \mid \neg R, S) = 1$, $P(W \mid R, \neg S) = 1$, $P(W \mid \neg R, \neg S) = 0.1$

Use the Bayesian Network above to answer the following questions, using inference by enumeration. Show your work.

(a) [2] What is the probability that there was a bird on the lawn?

(b) [3] If there was a bird on the lawn, what is the probability that the sprinkler was on?

(c) [3] If there was a bird on the lawn, what is the probability that the lawn was wet?

(d) [3] If there was a bird on the lawn, what is the probability that the lawn was dry, and it did not rain, and the sprinkler was not on?

(e) [3] What is the probability that the lawn was wet?

(f) [3] If the lawn was wet and the sprinkler was on, what is the probability that it rained?

(g) [3] Suppose we remove the connection between *S* and *B*, and then add a directed connection from *B* to *W*. Draw this new network. Considering *W* as the class variable, is this a Naïve Bayes network? Why or why not?

Problem 3. [65] Naïve Bayes Classifier in Biomedical Informatics

One interesting thing about Naïve Bayes is that it needs less training data than discriminative learning methods if conditional independence assumptions actually hold. Thus, Naive Bayes (NB) is one of the most effective and efficient classification algorithms that has been used for over 50 years in biomedical informatics (where collecting a large amount of data is a big problem). It is very efficient computationally and has often been shown to perform classification surprisingly well, even when compared to much more complex methods.

In this homework, you are to implement a general Naive Bayes classifier for heart attack detection using cardiac images. A SPECT (Single Proton Emission Computed Tomography) scan of the heart is a noninvasive nuclear imaging test. Doctors use SPECT images to diagnose coronary artery disease and to detect if a heart attack occurred.

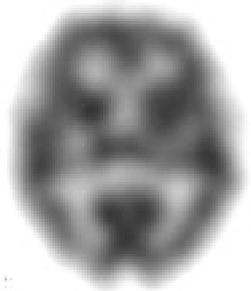


Figure 2. A cardiac SPECT

You will diagnose patients based on their cardiac SPECT images. Each patient will be classified into one of two categories: normal (negative) and abnormal (positive). Each SPECT image was pre-processed to extract multiple attributes. As a result, 22 binary attributes (aka features) were created for each patient from their SPECT image. Your task is to build a Naïve Bayes classifier based on this data, and then use it to predict if a patient is normal or not.

Naïve Bayes Classifier

- Binary Classification: Your program is intended for binary classification problems only (i.e., classify patients as negative or positive).
- Discrete values: All attributes are discrete-valued. Your program should be able to handle an arbitrary number of attributes with possibly different numbers of values for each attribute even though the supplied dataset has only binary attributes.
- 'Add one' smoothing: To avoid any "zero count" issues, use Laplace estimates (pseudocounts of 1) when estimating all probabilities. (See lecture notes for details.)
- Logs: Convert all probabilities to log probabilities to avoid underflow problems, using the natural logarithm (`log(x)` in Java).
- To break ties, classify as positive (abnormal) (to be conservative)

Hints

(1) "Add-1" Smoothing

Conditional probability:

$$P(X = x|Y = y) = \frac{\#(x, y) + 1}{\#(y) + m}$$

- #(x, y): number of instances having X = x and Y = y
- m: size of attribute X (number of distinct values in X)

Prior probability:

$$P(Y = y) = \frac{\#(y) + 1}{\#(instances) + n}$$

- #(y): number of instances having Y = y
- n: size of attribute Y (number of distinct values in Y)

(2) From Probabilities to Log Probabilities

To avoid underflow problems you should convert all probabilities to log probabilities. Use the natural logarithm (`log(x)` in Java), and apply the log function to all probabilities: calculate each probability $P(X|Y)$ or $P(Y)$ based on counting, then calculate the log value of each probability, and use them for further computation.

(3) Classify an Instance

You should compare posterior probabilities of all class values and choose the highest one. For example, the posterior probability of the positive class given *a* and *b* is:

$$\log\big(P(Y = pos|A = a, B = b)\big)$$
$$= \log\big(P(a|pos)\big) + \log\big(P(b|pos)\big) + \log\big(P(pos)\big) - \log(P(a, b))$$

Note: You may *not* need $P(a, b)$ because it's same for all class values given a particular instance.

Evaluation

For a binary classification problem, we usually care about precision and recall (https://en.wikipedia.org/wiki/Precision_and_recall). Your program should evaluate the classifier using the test set and three metrics: accuracy (the fraction of correctly classified instances), precision and recall.

I/O Format

- **Input:** Paths to train and test files:

```
java HW5 <train-set-file> <test-set-file>
```

- **Output**

Your program should output the following:

- One line for each instance in the test set (in the same order as this file) indicating (i) the index of the instance, (ii) the actual class, and (iii) the predicted class by NB. (*This is implemented by the function* `printOut` *in* `HW5.java`)
- Three lines of evaluation scores rounded to 3 digits after the decimal point (you can use the function `round3` in `HW5.java`):
    - Accuracy
    - Precision
    - Recall

Dataset

You can test the correctness of your code using the given dataset. Another dataset with the same format will also be used for grading. The SPECT dataset (in dataset folder) describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. This dataset was obtained from the UCI repository: `https://archive.ics.uci.edu/ml/datasets/SPECT+Heart`

Description:

- The **last** column is the class label: 1 for positive label (i.e., patient had a heart attack), 0 for negative label (patient is normal).
- The first row indicates the number of values for each attribute (in the same order)
- Other rows: Each row is a data instance including all attributes and the class.
- Data values have been transformed to integers (0, 1, 2, …). For example, if attribute f1 has 5 values, then the values of f1 are {0, 1, 2, 3, 4}.
- Training set has 80 instances (SPECT.train.csv)
- Testing set has 187 instances (SPECT.test.csv)

Skeleton Code

We have provided skeleton code containing the following four java files:

1. `Label.java:` defines 2 labels

2. `NBClassifier.java`: the interface of Naïve Bayes that has 2 methods:

    - `public void fit(int[][] data)`: fits the training dataset and learns parameters of the classifier
    - `public Label[] classify(int[][] instances)`: classifies new instances based on learned parameters

3. `NBClassifierImpl.java:` implements methods in the interface

4. `HW5.java` does these tasks:
    - Reads command line arguments
    - Loads the training set file and the test set file
    - Converts data values of datasets into integers (`int[][] array`)
    - Initializes a Naïve Bayes Classifier instance:
        - o Trains the classifier using the training set (`fit`)
        - o Classifies all instances in the test set to obtain the predicted class labels
        - o Extracts the actual (teacher) labels from the test set
        - o Calls the evaluation function to compare the predicted labels and the actual labels and computes the required evaluation scores
    - Prints output (all necessary print functions have been implemented)

You should implement the following three methods:

- `public void fit(int[][] data)` in `NBClassifierImpl.java`

- `public Label[] classify(int[][] instances)` in `NBClassifierImpl.java`

- `private static double[] evaluate(Label[] yTrue, Label[] yPred)` in `HW5.java`

Hand-In Instructions

Submit all your java files according to the instructions given above. You should *not* include any package paths or external libraries in your program.