

## Programming Workshop Five Naïve Bayesian Classifier

Student Name: \_\_\_\_\_

Student Name: \_\_\_\_\_

In this programming exercise, you will implement a simple classifier of documents by their content, for example into spam and non-spam E-mails. Both group members are expected to contribute equally to this project.

*Due date and exact submission guidelines will be posted here after the work on this workshop has started.*

### Program specifications:

1. Do not use Perl modules or packages written by someone else. Implement everything on your own. Any "borrowed code" submission will result in zero points.
2. You are allowed and encouraged to use any Perl built-in function.
3. All files must be in `pw5` directory.
4. Name the program `bayesian.pl`
5. The program must compile with the `strict` pragma.
6. The program must compile with `-w` option in DH450.
7. The program must have (at least) two subroutines: `train` and `classify`
8. The `classify` subroutine should run in one of the following two modes:
  - single-file mode: prompt the user for the name of the file to classify and output the following:  
`file_name class` (where `class` is `spam` or `nonspam`)
  - batch-classify: your classifier should iterate over all the files in the current directory whose names are in the following format: `unknownXX.txt`, where `XX` is a number (2 or more digits). For each file, the following line should be output:  
`file_name class` (where `class` is `spam` or `nonspam`)
9. The `train` subroutine will use the input files from two directories (located in the current directory): `spam` and `nonspam`. The naming conventions for files inside those directories are on-line (these are the same as for the files you e-mailed me). Chapters 10 and 11 are good reference.
10. The program should be well-documented. Header comments are required. `POD` is encouraged to try. Use `Statistics::Chisquare` module as an example.

## Background

Let  $D$  denote a document undergoing classification. Let  $C = \{\text{spam}, \text{nonspam}\}$  be a set of two possible classes  $D$  can belong to. Let  $\{w_1, w_2, \dots, w_n\}$  be a vector of words in a dictionary. We define the Bayesian posterior probability as follows:

$$p(C | D) = \max \begin{cases} p(C = \text{spam}) \prod_i p(w_i | C = \text{spam}) \\ p(C = \text{nonspam}) \prod_i p(w_i | C = \text{nonspam}) \end{cases}$$

Terms  $p(C = \text{spam})$  and  $p(C = \text{nonspam})$  are called prior probabilities.

In other words, we can compute two posterior probabilities and select the class that gives us the maximum posterior probability. This step is called classification. During this step, an unseen document is scanned word by word and for each word,  $p(w_i | C = \text{spam})$  is looked up and multiplied by the intermediate result. Once the end of the document is reached, the result is multiplied by the prior probability. Next, the process is repeated for nonspam posterior computation. The maximum of the two posteriors is chosen and a class is assigned to the document.

The training phase involves finding priors and  $p(w|C)$  from some training sets.

Recall two basic properties in probability:

1. Probability of an event  $A$  is  $0 \leq p(A) \leq 1$ .
2. The sum of probabilities of all possible outcomes must be 1.

Try the following example. Assume you are given two e-mail messages:

### **spam e-mail message:**

*Now you can get out of DEBT fast, and start to enjoy life agin. Simply complete our fast two minute online application, and help will be on its way too you within 48 hours, but usually the next business day!*

### **nonspam e-mail message:**

*Write a program that allows the user to submit a question. When a question is submitted, the server should choose a random response from a list of vague answers and return a new page displaying the answer. Implement the program as a Perl program using CGI.pm.*

1. Explain how you will create a dictionary of words for your classifier.

