Heidi Gebauer (geba), Markus Roos (roor)
School of Engineering T

A Spam Filter based on the Naive Bayes Approach: Exercise Set

Source: Andrew Ng, Stanford, CS229 Machine Learning

In this exercise you will implement a spam classifier using the naive Bayes approach. You are given a set of spam/non-spam mails, which have been preprocessed (e.g., mail/web addresses, currencies and numbers were replaced by generic tokens). The corresponding files can be found on olat.

Note: The use of these files is restricted to the purposes of this class. Please do not distribute the data to others!

Data sources: The spam mails are provided by Christian Shelton; the non-spam messages are taken from the newsgroup-data `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html`

Auxiliary Files (available on moodle)

Under Python, the auxiliary files are loaded with the routine `loadData` from file `load_data`. It returns a dict with the matrices, vectors and arrays mentioned below (key is the corresponding name).

- The files `spamSampleOriginal` and `spamSamplePreprocessed` illustrate how the preprocessing is applied.
- The file `tokensList` serves as a "dictionary": Every word corresponds to an id number (e.g., the word "affect" has the number 24). Simliarly, `tokenArray` stores the ith word at position i (e.g., tokenArray[23] gives "affect").
- `trainMatrix` stores a matrix, each row of which represents a message of the training set: Each entry of a row represents the number of occurrences of the corresponding word in the respective document. For example, the entry "3" in row 14 and column 18 denotes that word nr. 18 of the dictionary appears 3 times in document nr. 14.
- `testMatrix` has the same structure as `trainMatrix` and is used for testing.

- `trainLabels` represents an array consisting of 0/1 entries. The ith entry of this array indicates whether the ith document is spam (1 = spam, 0 = non-spam).
- `testLabels` has the same structure as `trainLabels` and is used for testing.

Task 1 For this task do not implement additional techniques (as Laplacian smoothing or logarithmization) yet.

(a) Write a script `train` that computes row-vectors `probTokensSpam` and `probTokensNonSpam` such that the ith entry denotes the probability that word i occurs given that a mail is spam and non-spam, respectively.

Test Cases: For the first test case ("trainMatrixEx1" etc) you should get [0.2 0.2 0.3333 0.0667 0.2] (spam) and [0.2857 0.1429 0.1429 0.1429 0.2857] (non-spam).

For the second test case ("trainMatrixEx2" etc) you should get [0.6 0.1 0.05 0.15 0.1] (spam) and [0.2 0.1 0.35 0.1 0.25] (non-spam).

(b) Write a script `test` that predicts the class label of each document of the test set using the naive Bayes approach. Store the found classifications in a boolean vector `output` ("1" stands for spam, "0" stands for non-spam) and determine also the error rate using the true labels stored in `testLabels`.

Test Cases: For the first test case you should get [1; 1; 1; 0; 1; 1] and an error rate of 0.3333.

For the second test case you should get [1; 1; 0; 0; 0; 0] and an error rate of 0.1667.

Task 2

Modify your scripts from the previous task such that Laplacian smoothing and appropriate logarithmization are applied.

Test Cases: The correct values for the test cases are listed below.

- First case: [0.2 0.2 0.3 0.1 0.2] (spam probabilities), [0.25 0.1667 0.1667 0.1667 0.25] (non-spam probabilities), [1; 1; 1; 1; 1; 1] (output), 0.5 (error rate)
- Second case: [0.52 0.12 0.08 0.16 0.12] (spam probabilities), [0.2 0.12 0.32 0.12 0.24] (non-spam probabilities), [1; 1; 0; 0; 0; 0] (output), 0.1667 (error rate)
- Third case: The files `probSpamEx3` and `probNonSpamEx3` contain the values for the spam and non-spam probabilities. The remaining results are [0; 0; 0; 0; 0; 1; 1; 1; 1; 1] (output) and 0 (error rate).

Optional Task: It seems that some particular tokens are a very good spam-indicator. We use the measure

$$m(i) := \frac{P(\text{token } i \text{ occurs}|\text{mail is spam})}{P(\text{token } i \text{ occurs}|\text{mail is nonspam})}$$

to determine the quality of an indicator. Determine the 5 tokens having the hightest value for this measure.

Hint 1: You can reuse the results from Task 2.
Hint 2: To output the $i$the word of the dictionary, use `tokenArray`.