CSE446: Homework #3

May 22, 2015

1 General Instructions

- Please submit **both your report and your code**. Include instructions for compilation and usage.
- You can use **Python, C/C++, Java, R, Rust.**
- If you use Java, use it with Eclipse. Please submit your whole project folder so that I am able to import it as a project right away.
- C/C++, Python and R implementations will be tested on the **attu** server, so make sure they can be compiled and run with no issues. If it is absolutely necessary to use some package that is not available there, state exactly what you are using and how to install it.
- Test your code before submission so that you make sure its running. Double-check that your files were successfully submitted and are in the correct form.
- Double-check that the results of your code match your report.
- Do not submit data. Your implementation should read the data from a specific data folder (empty) in your submission and should be working regardless of the machine (so don't link the code to your personal folders on your personal machine). The data file names are not to be given as an argument. Hardcode them to be what they are in the datasets.
- If you are unsure about anything visit us during office hours. You can avoid a lot of wasted effort.
- If you know something is wrong with your work, discuss what it is.
- **Start early!**

2 Naive Bayes Classifier

2.1 Questions

1. The dataset we will be using is a subset of 2005 TREC Public Spam Corpus, containing 9000 training examples and 1000 test examples. You can download it here. Each line in the train/test files represents a single email with the following space-delimited properties: the first is the email ID (in the form /xxx/yyy), the second is whether it is 'spam' or 'ham' (non-spam), and the rest are words followed by their occurrence numbers. (Note that numbers may be words, so don't worry if a line contains multiple numbers in a row). The data has been pre-processed to remove non-word characters (e.g. '!') and to select features similar to what Mehran Sahami did in his original paper, though with larger cut-offs since our corpus is larger.

2. Using the training data, compute the prior probabilities $P(spam)$ and $P(ham)$. What is $P(spam)$?

3. Determine the vocabulary and compute the conditional probabilities $P(w_i|spam)$ and $P(w_i|ham)$ using the m-estimate discussed in class (with $m = |Vocabulary|$ and $p = 1/|Vocabulary|$). In this context we consider each word as a training example, so n is the total number of words (in either ham or spam documents) and n_c is the number of times w_i appeared in those documents (including multiple occurences in the same email). What are the 5 most likely words given that a document is spam? What are the 5 most likely words given that a document is ham?

4. Use these probabilities to classify the test data and report the accuracy (i.e. the percentage of correct classifications). Note that directly computing $P(spam|w_1, \ldots, w_n)$ and $P(ham|w_1, \ldots, w_n)$ can cause numerical precision issues, since the unnormalized probabilities are very small (i.e. the numerator in Bayes' theorem). Instead, you should compare the log-probabilities of being ham/spam.

5. Vary the m parameter, using $m = |Vocabulary| \times [1, 10, 100, 1000, 10000]$ and plot the accuracies vs. m. What assumptions are we making when the value of m is very large vs. very small? How does this affect the test accuracy?

6. If you were a spammer, how would you modify your emails to beat the classifiers we have learned above?

7. **Extra-credit:** Our feature selection makes learning much easier, but it also throws out useful information. For example, an exclamation mark (!) often occurs in spam. Even the format of email sender matters: in the case when an email address appears in the address book, a typical email client will replace it with the contact name, which means that the email is unlikely to be a spam (unless, of course, you are a friend of the spammer!). Sahami's paper talked about a few such features he had used in his classifier. For extra credit, you can play with the original files and come up with useful features that improve your classifier. Here are the lists of the files used in train/test.

2.2 Your submission

Your report:

- A high-level description on how your code works.
- The accuracies you obtain.
- If all your accuracies are low, tell us what you have tried to improve and what you suspect is failing.

Your code:

- See general instructions.

3 Neural Networks

Problems 4.2 and 4.8 from Mitchell.

4 Ensemble Methods

Implement the bagging algorithm on top of your decision tree learner from Assignment-1. You should make your code as modular and learner-agnostic as possible. In other words, the main module of Bagging should treat the base learner as a black-box and communicate with it via a generic interface that inputs a set of instances and receives a classifier. The module should at each iteration sample with-replacement to generate a new training set of equal size and store the corresponding learnt classifier. The classification output of the ensemble is then just the majority vote on this set of learned classifiers.

Dataset: You'll be using the Letter recognition *dataset* from earlier in the course for testing your classifier. The archive (as before) contains the files `features.txt` & `labels.txt`; the latter encodes the alphabet as integers in the range $0, \ldots, 25$ (there is only one case). Use the first 12000 samples as your training set, and the latter 8000 as the test set.

Questions: Using conditional entropy (or equivalently, information gain) as the evaluation criterion answer the following,

i. Give a high-level description of how your code works, and submit your code.

ii. For each $p \in \{1.0, 0.05, 0.01\}$ as argument for the ID3 classifier, **plot** separately, the training & test accuracies for $1, 2, 3, \ldots 40$ samplings (three graphs in total). Also include in each plot, the accuracies obtained by ID3 on the full training set without sampling.

iii. Is Bagging useful when $p \neq 1.0$? Explain.