SPAM CLASSIFICATION

To demonstrate naïve bayes classification, we have implemented a classifier that can classify between spam and non spam messages.

To train the naïve bayes classifier we have used 1000 training messages, and to predict how accurate the classifier is, we have used 1000 test messages also. The result shows that classifier is able to classify 946 messages out of these 1000 correctly.

Here we are training a classifier to classify whether a given email, x, is spam (y = 1) or non-spam (y = 0). In particular, we need to convert each email into a feature vector n dimensional feature vector. To construct the feature vector we have to first preprocess the email.

**Sample email**

> Anyone knows how much it costs to host a web portal ?
>
Well, it depends on how many visitors youre expecting. This can be anywhere from less than 10 bucks a month to a couple of $100. You should checkout http://www.rackspace.com/ or perhaps Amazon EC2 if youre running something big..
To unsubscribe yourself from this mailing list, send an email to:
groupname-unsubscribe@egroups.com

Sample email that contains a URL, an email address (at the end), numbers, and dollar amounts. While many emails would contain similar types of entities (e.g., numbers, other URLs, or other email addresses), the specific entities (e.g., the specific URL or specific dollar amount) will be different in almost every email. Therefore, one method often employed in processing emails is to "normalize" these values, so that all URLs are treated the same, all numbers are treated the same, etc. For example we could replace each URL in the email with the unique string "httpaddr" to indicate that a URL was present.

Preprocessing steps we have implemented:

- Lower-casing: The entire email is converted into lower case, so that capitalization is ignored (e.g., IndIcaTE is treated the same as Indicate).
- Stripping HTML: All HTML tags are removed from the emails. Many emails often come with HTML formatting; we remove all the HTML tags, so that only the content remains.
- Normalizing URLs: All URLs are replaced with the text "httpaddr".
- Normalizing Email Addresses: All email addresses are replaced with the text "emailaddr".
- Normalizing Numbers: All numbers are replaced with the text"number".
- Normalizing Dollars: All dollar signs ($) are replaced with the text "dollar".

• Word Stemming: Words are reduced to their stemmed form. For ex- ample, “discount”, “discounts”, “discounted” and “discounting” are all replaced with “discount”. Sometimes, the Stemmer actually strips off additional characters from the end, so “include”, “includes”, “included”, and “including” are all replaced with “includ”.
• Removal of non-words: Non-words and punctuation have been re- moved. All white spaces (tabs, newlines, spaces) have all been trimmed to a single space character.

**Preprocessed sample email**

| |
|---|
| anyon know how much it cost to host a web portal well it depend on how mani visitor your expect thi can be anywher from less than number buck a month to a coupl of dollarnumb you should checkout httpaddr or perhap amazon ecnumb if your run someth big to unsubscrib yourself from thi mail list send an email to emailaddr |

After preprocessing the emails, we have a list of words for each email. The next step is to choose which words we would like to use in our classifier and which we would want to leave out. For this exercise, we have chosen only the most frequently occuring words as our set of words considered (the vocabulary list). Since words that occur rarely in the training set are only in a few emails, they might cause the model to overfit our training set. The complete vocabulary list is in the file vocab.txt. Our vocabulary list was selected by choosing all words which occur at least a 100 times in the spam corpus, resulting in a list of 1899 words. In practice, a vocabulary list with about 10,000 to 50,000 words is often used.

Given the vocabulary list, we can now map each word in the preprocessed emails into a list of word indices that contains the index of the word in the vocabulary list.

To implement the feature extraction that converts each email into a vector in Rn. For this exercise, we will be using n = # words in vocabulary list. Specifically, the feature $x_i \in \{0, 1\}$ for an email corresponds to whether the i-th word in the dictionary occurs in the email. That is, $x_i = 1$ if the i-th word is in the email and $x_i = 0$ if the i-th word is not present in the email. So now we have feature vector for each email message. We can apply naïve bayes algorithm on this and to classification.