Project 4: SVM Spam Classifier

Module:
Machine learning (IMC-4302C)
ESIEE

Instructors:
Adriana GOGONEL
Slim BEN AMOR

1 Basic notions about machine learning

1. Give the steps of the algorithm of best subset selection.
2. What is the difference between supervised and unsupervised learning?
3. Describe a real-life situation in which linear regression might be useful (specify the features, design the predictor) and transforming the predictor logistic regression might be also used. Describe that transformation of the predictor.

2 Programming Section

In this project, you are invited to work with **spam email dataset** from Apache SpamAssassin Project. First, you should train an SVM classifier using "spam_train.txt" file in github. This file contains 1899 features per email (line). Each features represent the number of occurrence of a given word from "vocab_list.txt" file. In fact, raw emails from SpamAssassin project are preprocessed to substitute or remove some expression and punctuation. Then, we select the most occurring words in the resulting emails to build a vocabulary list. In our case, we select all words which occur at least a 100 times in all emails. This results in a list of 1899 words. In practice, a vocabulary list with about 10 000 to 50 000 words is often used.

After training the SVM Classifier, you should test it on raw emails. You have in github folder some file containing email samples. You could also use some email from your mailbox. You should preprocess these raw emails to extract features. Thus, you could feed the features vector to your trained classifier that predict if it is spam email or not. Finally, you should evaluate the generalized performance of your spam classifier on the given test set.

Preprocessing Emails

In general emails contain different types of entities (e.g. numbers, dollar amount, URLs, or other email addresses). These entities 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. This has the effect of letting the spam classifier make a classification decision based on whether any URL was present, rather than whether a specific URL was present. This typically improves the performance of a spam classifier, since spammers often randomize the URLs, and thus the odds of seeing any particular URL again in a new piece of spam is very small.

Below needed preprocessing and normalization steps are enumerated. Besides, some regular expressions and function usefull for these processing are given:

- **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. You should remove all the HTML tags by replacing this regular expression `'<[^<>]+>'` with white space. You could use "sub()" function from regular expression library in python.
- **Normalizing URLs:** All URLs are normalized by replacing this regular expression `'(http|https)://[^\s]*'` with the text 'httpaddr'.
- **Normalizing Email Addresses:** All email addresses are normalized by replacing this regular expression `'[^\s]+@[^\s]+'` with the text 'emailaddr'.
- **Normalizing Numbers:** All numbers are normalized by replacing this regular expression `'[0-9]+'` with the text 'number'.
- **Normalizing Dollars:** All dollar signs ($) are replaced with the text dollar (find the appropriate regular expression to use).
- **Removal of non-words:** Non-words and punctuation should be removed and all white spaces (tabs, newlines, spaces) have to be trimmed to a single space character by replacing the following regular expression with a single white space.

  ```
  '[@/#.\-:\[\]&*+=?!(){},\'\'">_<;% \t\n\r]+'
  ```

  Concerning the leading and trailing space you could removed them using "strip()" function.
- **Word Stemming:** Words are reduced to their stemmed form. For example, 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. You may use PorterStemmer class from nltk.stem module. For that you need to download "punkt" package using this command in python: `nltk.download('punkt')`

Questions

For each of these questions, you should explain which method or steps are used and justify your choice/strategy. You should also discuss and interpret all the results you get. You are encouraged to put some notes and remarks directly in the notebook within a markdown cell.

1. Load the dataset from "spam_train.txt" file in github and explore it. Try to solve the problem of missing value if any.
2. Train an SVM classifier using loaded training data.
3. Introduce the regularization on your model if it wasnt used and tune its regularization parameter.
4. Implement the preprocessing steps described above. Try to use given regular expressions and functions that may help. Then, you should split the email with white space character in order to get a list of words. Thus, you could apply stemming word by word. However, you need to remove any non-alphanumeric character from each word by replacing this regular expression `'[^a-zA-Z0-9]'` with empty string.
5. Load the vocabulary list ("vocab_list.txt" file in github) and extract features from the processed email. For that you should count the occurrence in the email of each word in the vocabulary list. Then, put this count in a vector of 1899 elements according to the order of word in the vocabulary list. For example, if the 3rd element of the resulting features vector is 5 this means that the 3rd word in the vocabulary list was encountered 5 times in the processed email.
6. Estimate the generalized performance of this model using adequate metrics..