CS5691: Pattern Recognition and Machine Learning Assignment

K Kamalesh Kumar
CE20B054

May 3, 2023

This report deals with the problem statement of spam classification. The algorithm used for the task is the Naïve Bayes algorithm, which assumes class conditional independence between the features given the class. The dataset used for this purpose is the Enron-Spam dataset (available at `http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/index.html`) which contains 367 text emails for non-spam category and 1500 spam emails. The dataset is split into 3166/1235 emails for ham/spam respectively for training, and 506/265 for ham/spam for testing.

Standard text preprocessing is done over the dataset, like converting all words to lowercase, removing stop words and punctuations, and also stemming each word into its root word (learned or learnt or learning will all become learn). Finally, each mail is represented as a token (or list) of words after all the preprocessing. For the Naïve Bayes algorithm, a dictionary containing all the tokens for a given category is maintained for both of the classes, which also contains the count of the number of times that word has occurred for example, in the set of all spam mails in the training set and similarly for the ham case. Using these counts, the probability of each word given the class is calculated, and similarly the probabilities of the spam and the ham classes are as follows:

$$P(X_i|\text{class } c) = \frac{\#\text{ of times } X_i \text{ occurred in } c}{\text{Sum of counts of all words in } c}$$
$$P(\text{class } c) = \frac{\#\text{ emails of class } c}{\#\text{ of emails in the training data}}$$

Note that in the above dictionary model, only words present in the training data are present, so to tackle new words present during inference, an additional keyword "UNKNOWN_WORD" is introduced with a count of 1 (similar to Laplace Smoothing) in both classes. Thus during inference, a given email's is preprocessed and tokenized, and the probability of each class given the email is computed using Bayes' rule, and whichever class has the highest probability, the email is classified as it. If the word during testing is not present in the stored dictionary, the probability of the "UNKNOWN_WORD" is used. To tackle numerical underflows, which are bound to occur since the actual probabilities are found by taking the product across all the words in the given email, instead the logarithm of the probabilities is considered (since it would take the sum).

Running inference on the test set gave a test accuracy of 94% for the spam category and 94.9% for the ham category, which is indeed a great performance given the simplicity of the Naïve Bayes algorithm.

1 Instructions for running inference

For testing the code and algorithm, kindly run the `inference.py` python file. Replace the `test_dir` variable with the folder containing the test emails (kindly ensure they only contain emails and not any other subfolder.) The dictionary containing the words and their respective probabilities for each class is stored in `p_word_given_spam` and `p_word_given_ham` and marginal class probabilities are stored in `p_class` (all these values are found during training and will be used during testing). If you would like to train the model, run `main.py`, otherwise use the already submitted probabilities for testing. After the inference code completes, the predicted categories are stored in the `final_predictions.txt` file.

Libraries that are used: nltk (for text preprocessing, use pip install nltk), re, pickle, etc.

Note: While running inference, if any email text file contains non-English or non-decodable characters, these files will face Unicode Decode Error while trying to read the file, and thus they will be skipped, and the loop will continue to the next file.