CS5691 Pattern Recognition and Machine Learning

Assignment 3

Ruthwik Chivukula

ME21B166

Contents

1 Problem Statement **2**

2 Dataset **2**

3 Data preprocessing and feature extraction **2**

4 Algorithm **3**

4.1 Support Vector Machines 3

4.2 Naive Bayes Classifier 5

5 Code and Conclusion **6**

1 Problem Statement

In this assignment, we have been asked to propose and build a model to detect spam/ham emails. This begins right from choosing the dataset for training, the feature extraction and data preprocessing steps. Followed by testing out different algorithms that have been covered in this course. I have tried Support Vector Machines and Naive Bayes algorithms to solve this classification problem. The methods adopted are clearly specified under each of the following sub-sections.

2 Dataset

I have chosen a dataset from Kaggle for this problem statement. The link is attached below. The dataset has two columns, one containing the email text and the other containing the corresponding label. The dataset has a total of 5695 emails, which have been classified as spam or ham. The dataset is imbalanced, with 24% spam and the remaining 76 % being ham.

https://www.kaggle.com/datasets/jackksoncsie/spam-email-dataset/data

3 Data preprocessing and feature extraction

1. Downloaded the CSV file and performed preliminary preprocessing steps using the Pandas library.
2. Removed punctuation from the text and made all characters lowercase.
3. Used NLTK library for lemmatization, reducing all the words to their root form.

4. Created a new column in the data frame containing the list of the processed words that have occurred in the corresponding data samples.

5. Prepared Vocabulary for the dataset by compiling all the unique words that have occurred in the dataset.

6. Prepared Bag of Words embedding from scratch and added it as a new column to the data frame.

7. Vectorized the text into an array of zeros and ones with dimensions equal to the vocabulary length.

4 Algorithm

I have tested Support Vector Machines and Naive Bayes algorithms on the dataset. The specifics are mentioned below.

4.1 Support Vector Machines

The optimization problem is formulated as follows:

$$\min_{\omega,\epsilon} \frac{1}{2}\|\mathbf{w}\|^2 + c\sum_{i=1}^{n}\epsilon_i \tag{1}$$

subject to the constraints:

$$w^T x_i y_i + \epsilon_i \geq 1 \quad \text{for all } i \tag{2}$$

$$\epsilon_i > 0 \quad \text{for all } i \tag{3}$$

Equation (1) represents the objective function, and Equations (2) and (3) represent the constraints.

The above constraint optimization is solved using Lagrange multipliers. We formulate a dual problem solving in which we obtain the desired solution.

$$\max_{\alpha\in[0,C]} \alpha^T\mathbf{1} - \frac{1}{2}\alpha^T Y^T X^T X Y \alpha \tag{4}$$

This can be solved by a simple projected gradient descent over alpha.

The above equation is a soft-margin Support Vector Classifier, which introduces a ϵ term, helping to waive off the strong linear separability condition. The multiplied c coefficient is a scaling factor that determines the cost of a bribe. (bribe refers to ϵ here, an analogy discussed in class)

I have imported the Sklearn implementation of the Support Vector Classifier with the kernel type set as linear. The heavy math involved is done behind the scenes in the Sklearn model, making it convenient for users to train and fit it.

I set the train test split as 65:35. The SVC model secures 97.94% accuracy on the test data.

4.2 Naive Bayes Classifier

Implemented Bernoulli Naive Bayes algorithm from scratch. From the Bayes rule, we know the following result:

$$P(y^{test} = 1|X^{test}) = \frac{P(y^{test} = 1)P(X^{test}|y^{test} = 1)}{P(X^{test})} \tag{5}$$

so given a test point, if $P(y^{test} = 1|X^{test}) > P(y^{test} = 0|X^{test})$, then we predict the class to be 1 else we predict 0.

The number of parameters involved in defining a Naive Bayes algorithm are:
$p,\ \ p_1^1, p_2^1, p_3^1, ..., p_d^1\ \ ,\ \ p_1^0, p_2^0, p_3^0, ..., p_d^0$ which are 2d + 1 parameters.

$$\hat{p} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{6}$$

$$\hat{p}_j^y = \frac{\sum_{i=1}^{n} I(f_j^i = 1; y_i = y)}{\sum_{i=1}^{n} I(y_i = y)} \tag{7}$$

where j is the jth word and y is the label (either 1 or 0)

using the above equations we can write the following result:

$$P(y^{test} = 1|X^{test}) = \prod_{k=1}^{d} (\hat{p}_k^1)^{f_k}(1 - (\hat{p}_k^1)^{1-f_k}\hat{p} \tag{8}$$

$$P(y^{test} = 0|X^{test}) = \prod_{k=1}^{d} (\hat{p}_k^0)^{f_k}(1 - (\hat{p}_k^0)^{1-f_k}(1 - \hat{p}) \tag{9}$$

equation (8) and equation (9) will have a normalizing term in the denominator, which I have ignored as we are concerned about relative value. whichever is great, the prediction will be of that class.

Naive Bayes achieves 97.5% accuracy on the training dataset. It has to be noticed that there is no training involved in Naive Bayes. With the help of the trainset, we compute the probability values. Then, using the Bayes rule, we calculate the class probability on the test set using the probability values of the 2d+1 parameters we obtained. This algorithm assumes conditional independence due to which the algorithm is naive and easy to use.

5 Code and Conclusion

The code implementations, along with the inference code, have been attached in the zip file. The inference code is a .py file that can take a txt file as input and predict whether the email is spam or ham. Both models boast high accuracy for the email spam detection problem statement.