B(E)3M33UI Semestral project 1: Spam filter

Sinchiguano Cesar

***Abstract*— This paper explores and identifies the use of four different learning algorithms (Logistic Regression classifier, AdaBoost classifier, Random Forest classifier and Naive Bayesian Classifier) for classifying spam messages(spam filtering) from a data set. A comparative analysis of the algorithms has also been presented and it will showcase that simple AdaBoost classifier taking from the ensemble models, could be efficient for the dataset which could be classified as binary tree.**

I. Assignment

The goal of this semestral task is to employ the knowledge of the ML lectures and previous exercises, and apply it to a text classification task, namely called spam filtering. A dataset with text messages and their classification as spam or ham is provided, in addition to a skeleton code where it shows the use of a dummy classifier as the starting point.

II. Introduction

The classification algorithms such as Logistic Regression classifier, AdaBoost classifier, Random Forest classifier and Naive Bayesian Classifier are currently used in various datasets and showing a good classification result.

In this paper, the four algorithms already mentioned above are proposed as anti-spam filtering methods to compare their performances. For each algorithm, we develop it by adjusting some parameters better-called hyper-parameters to achieve its best possible predicted result with the use of Grid-search method. Grid-search is a way to select the best of a family of models, parametrized by a grid of parameters.

As to the data set. It was already divided into training and testing sets. The way as it was dividing our data set is called cross-validation. And it is a good one approach in order to avoid the overfitting state. We then trained each one of the algorithms using the training set, and as for the last step, we proceeded to evaluate their performances on the testing set. For the sake of the working algorithm, the training and testing sets had to be expressed as a vector. The dimensions in the vector are corresponding to the word frequency existing in the document. To achieve that, we used the CountVectorizer method as the previous step to the training one. The CountVectorizer method converts from a text document to a matrix of token counts. The resulting matrix is composed of a number of each word appears in the text document. Therefore, we transformed our given inputs of a text domain into a numerical domain which can be treated as task classification using each on of the classification methods.

1) **Logistic Regression classifier**
   Given a set of inputs X, we want to assign them to one of two possible categories (spam or ham). Logistic regression models the probability that each input belongs to a particular category. A function takes inputs and returns outputs. To generate probabilities, logistic regression uses a function that gives outputs between 0 and 1 for all values of X. There are many functions that meet this description, but the used in this case is the logistic function (sigmoid).

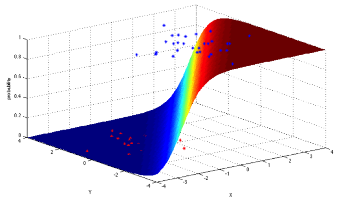

Fig. 1. An example of logistic regression. [1]

2) **AdaBoost classifier**
   AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. These are models that achieve accuracy just above random chance on a classification problem.
   The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps.
   Each instance in the training dataset is weighted. The initial weight is set to:

$$weight(x_i) = \frac{1}{n}$$

   Where x_i is the ith training instance and n is the number of training instances.
3) **Random Forest classifier**
   The random forest is another ensemble approach we used in this paper. The random-forest algorithm brings extra randomness into the model, when it is growing the trees. Instead of searching for the best feature while splitting a node, it searches for the best feature among a random subset of features. This process creates a wide diversity, which generally results in a better model.

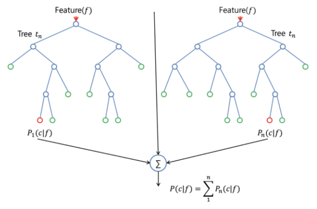

Fig. 2. An example of random forest with two trees. [2]

4) **Naive Bayesian (NB) Classifier**
Naive Bayesian algorithm is one of the frequently used machine learning methods for text categorization. The original idea of identifying whether an email is spam or ham looking at which words are found in the message and which words are absent from it. This approach begins by studying the content of a large collection of emails which have already been classified as spam or ham email. Then when a new email comes into some user's mailbox, the information obtained from the training set is used to compute the probability that the email is spam or ham from the words in the email. Given a feature vector $x = \{x_1, ..., x_n\}$ where are the values of attributes $X_1, ..., X_n$, and n is the number of attributes in the corpus. Let C denote the category to be predicted, i.e., $C \in \{spam, ham\}$. It uses a discriminate function to compute the conditional probabilities of $P(C_i|X)$. Here, given the inputs, $P(C_i|X)$ denotes the probability that, example X belongs to class C_i :

$$P(C_i|X) = \frac{P(C_i) * P(X|C_i)}{P(X)}$$

$P(C_i)$ is the probability of observing class i. $P(X|C_i)$ denotes the probability of observing the example, given class C_i. $P(X)$ is the probability of the input, which is independent of the classes.

III. Methodology

1) **Automatic tuning via grid search**
All algorithms implemented have hyper-parameters. A hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training. When using any type of the classifier described above, default parameters are taking into account. However, these default parameters often times are not the right choice for optimal prediction of the algorithm. So that, we were obliged to change most of them, or at least the most important, those who can be suitable for the spam filtering task. There are two ways to do so, one of them can be done by hand, which can be a time-consuming task, the other alternative is to do it through an optimizer better known in Machine learning world as grid search where the algorithm is optimized by cross-validation based on a scoring function. The performance of the selected hyper-parameters and trained model is then measured on a dedicated evaluation set that was not used during the model selection step.
2) **Model diagnostics**
The modified accuracy macc is used to measure the quality of the several algorithms for the spam filtering task on a certain dataset. In addition to additional model evaluations such as Receiver operating characteristic (ROC) and AUC (which stands for Area under the curve) are discussed in order to better understand which spam filtering is performing well or not.

IV. Experiments

In this section, the four classification algorithms are evaluated the effects based on different hyper-parameters and also different feature sizes in order to find the optimal model for spam filtering. Afterward, the performance of the spam filtering is measured in terms of accuracy. It means that each one of the filters is evaluated using a modified accuracy score which is already implemented in the supplied filter.py module as the function called $modified\_accuracy$ which uses the confusion matrix feature to compute TP, FP, TN, FN.

$$macc = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{TN} + 10 * n_{FP} + n_{FN}}$$

The algorithms described in the instroduction part, were carefully selected, not before a thorough research in many related jobs those who were done many years ago. Even so, in order to get a better and wide insight into the performance of each one of the models, we proceeded to run them with their default values, the results are shown in the Table I.

Classifier	Accuracy score
Logistic regression	70.75 %
Adaboost	70.75 %
Random Forest	64.39 %
Naive Bayesian	78.78 %

TABLE I
TEST ACCURACY OF DIFFERENT CLASSIFIERS WITH STANDARD PARAMETER-SETTINGS

Table I shows good performances for a Logistic regression, Adaboost and Naive Bayesian. On the contrary, Random forest performed badly as spam filtering, with that we confirmed the criterion of many researchers that Random Forest is not suitable for filtering spam messages and also that Naive Bayesian is more acute for the task. From now on, Random Forest classifier is irrelevant for a deeply study. Therefore, it is not dicussed anymore.

Considering as a first experiment the one we did above, running our model with default parameter, then we proceeded as a second experiment to run each model with the use an optimizer called grid search in order to find the best hyper-parameters for itself.

1) Logistic Regression classifier
   a) Grid Search
   We came out with four parameters: penalty, c,

and class weight as parameters for the logistic regression classifier and $ngram\_range$ as the only one parameter for the count vectorizer method. The reason to work with these exclusive parameters is that they are relevant in order to get a good result. We could use as many parameters as we wanted but the more we use them, the more time-consuming would be for the optimizer to find the best match. But it can be done if we could have worked with GPU which is suitable due to its high speed in processing data.

b) Filter Evaluation
As we already mentioned in the methodology part, the filter is evaluated using the modified accuracy score. The results are shown in the Table VI

$Data$	Default values	Tuned values
Training	1.0	0.99167
Testing	0.7076	0.8571

TABLE II
ACCURACY SCORE

Table VI shows clearly that there is a significant improvement in the performance of the model. The table shows both scores, the one with default parameters and the other with the parameters found by the optimizer.

c) Best parameters

$class\_weight$	balanced
penalty	l2
c	0.1
$ngram\_range$	(1, 3)

TABLE III
HYPER-PARAMETERS

Table III shows the final value of the hyper-parameters, those who were encounter by the grid search method.

2) AdaBoost classifier

a) Grid Search
We came out with two parameters: $n\_estimator$ for the AdaBoost classifier and $ngram\_range$ as a parameter for the count vectorizer method.

b) Filter Evaluation
As we already mentioned in the methodology part, the filter is evaluated using the modified accuracy score. The results are shown in the Table VI

$Data$	Default values	Tuned values
Training	1.0	0.99167
Testing	0.7075	0.83333

TABLE IV
ACCURACY SCORE

Table VI shows significant improvement in the perfomance of the model. The table also shows both scores, the one with default parameters and the other with the parameter found by the optimizer.

c) Best parameters

$n\_estimator$	30
$ngram\_range$	(1, 2)

TABLE V
HYPER-PARAMETERS

Table V shows the final value of the hyper-parameters, those who were encounter by the grid search method.

3) Naive Bayesian Classifier
We select the MultinomialNB, which is one of the two classic naive Bayes variants used in text classification.

a) Grid Search
We came out with two parameters: alpha for Naive Bayesian classifier and $ngram\_range$ as a parameter for the count vectorizer method. The alpha determines how much the data should be smoothed and the $ngram\_range$ decides between the following options: unigrams and $bi-grams$ in the model, in order to capture important information involving multiple tokens.

b) Filter Evaluation
As we already mentioned in the methodology part, the filter is evaluated using the modified accuracy score. The results are shown in the Table VI

$Data$	Default values	Tuned values
Training	1.0	0.99167
Testing	0.7878	0.7988

TABLE VI
ACCURACY SCORE

Table VI shows slightly improvement. It was expected to work well enough for this kind of task due to its usage in many task related to spam filtering.

c) Best parameters

$n\_estimator$	30
$ngram\_range$	(1, 2)

TABLE VII
HYPER-PARAMETERS

Table VII shows the final values of the hyper-parameters, those who were encounter by the grid search method.

V. Discussion

After the two experiments we did try on each one of the models. We are not able to tell which one should be selected, but it is clearly so far that logistic regression classifier performed well compare to the Adaboost classifier and the Naive Bayesian classifier. In order to take the final decision, we plot a Receiver operating characteristic curve better called ROC curve and we show the Area under the curve better known as AUC for each one of them.

1) Receiver Operating Characteristic curve
   In order to choose the optimal model, we have to look carefully which curve is closer to the upper left corner of the fig. 3 and simultaneously we have to see which one is with the less false positive rate. And finally we came out with the solution that the logistic regression is the one who meets the requirements. In Figure 3 , we can also see the values of the area under the curve for each model in order to better confirm the selected model(logistic regression) as final spam filtering.

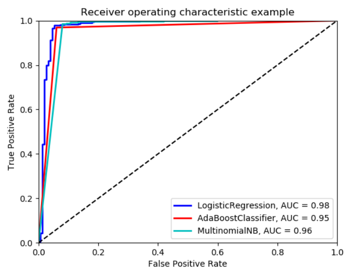

Fig. 3. Receiver Operating Characteristic curve (ROC curve)

VI. Conclusion

We investigate thoroughly the performance of the Naive Bayesian filter on a publicly available corpus, contributing towards standard benchmarks. At the same time, we compare the performance of the Naive Bayesian filter to an alternative memory-based learning approach, after introducing suitable cost-sensitive evaluation measures. Both methods achieve very accurate spam filtering, outperforming clearly the keyword-based filter of a widely used e-mail reader.
In general, hyperparameters are very specific to the type of machine learning mode you are trying to optimize. Sometimes, a setting is modeled as a hyperparameter because is not appropriate to learn it from the training set. A classic example are settings that control the capacity of a model(the spectrum of functions that the model can represent). If a machine learning algorithm learns those settings directly from the training set, then it is likely to try to maximize them which will cause the model to overfit(poor generalization).

References

[1] https://gab41.lab41.org/the-10-algorithms-machine-learning-engineers-need-to-know-f4bb63f5b2fa
[2] https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd