CS 191 Support Vector Machine Classifier Exercise Report

Submitted by
Femo Bayani and Mikaela Ramos

INTRODUCTION

Spam Emails

Spam emails are “unsolicited bulk e-mail” that are posted blindly to thousands of recipients. These spam emails can waste time and bandwidth, and “may expose under-aged recipients to unsuitable (e.g. pornographic) content” [1].

Support Vector Machine (SVM) Classifier

Support Vector Machine (SVM) is another supervised machine learning algorithm which defines its classifier with the use of hyperplanes. In this algorithm, different classes are separated using lines or hyperplanes (in the context of multidimensional space) [2].

There are several tuning parameters in SVM:

1. Kernel
2. Regularization
3. Gamma
4. Margin

The kernel can be set to three types: linear, polynomial, and exponential. Linear kernel is more commonly used whereas polynomial and exponential are used for calculating separation line in higher dimension. The calculation to be used for predicting the classifier of an input varies depending on the kernel.

The regularization parameter, on the other hand, is used to adjust how big the margin of the hyperplanes would be. The higher regularization parameter the smaller the margin hyperplane optimization will choose.

The gamma parameter dictates whether or not the points farther from the plausible line would also be considered in the calculation for the classifier. The lower the gamma the more farther points to be included in the calculation.

Lastly, the margin parameter sets how much the distance of the separating line would be from the nearest points of the classes it is separating. Larger margin is more desirable because smaller margin can result to overfitting.

IMPLEMENTATION

Data Preparation

The data set used is the TREC 2007 Public Spam Corpus, a dataset containing 75419 emails, 50199 (67%) of which are spam, and the remaining 25220 (33%) are ham.

The goal for data preparation of the researchers were to create a single `.csv` file with all the relevant preprocessed information stored. Multiprocessing was also utilized to speed up the preprocessing of the huge data set.

The following tasks were performed on the data set for data preparation and preprocessing using `preprocess.py` (a Python script, also in the source code and Github repository), based on the full subcorpora (index found at `dataset\trec07p\full\index`):

1. Reading the Files

   Each entry in the index was read, also followed by the reading of the corresponding email indicated in the entry, which undergo the next steps.

2. Obtaining the Body of the Email

   The body of each email is obtained and is passed to the next step for the actual preprocessing.

3. Preprocessing the Text
   a. Applying a lowercase filter

      All text is converted to lowercase.

   b. Stripping all the HTML tags

      All HTML tags are stripped, retaining only the actual text contents of the email. There have been cases where emails only contained images (through IMG tags), and thus these emails end up having no text left. Emails

   c. Removing accents

      All accented characters are transformed to their non-accented versions.

   d. Expanding contractions

      Contractions and slang words are transformed to their expanded forms. For example:

      ```
      you're -> you are
      i'm    -> I am
      ```

```
ima    -> I am going to
yall   -> you all
gotta  -> got to
```

e. Removing all special characters

All HTML tags are stripped, retaining only the actual text contents of the email. There have been cases where emails only contained images (through IMG tags), and thus these emails end up having no text left. These emails get removed from the data set later on.

f. Stop word removal

Stop words refer to the most common words in a language. This project uses NLTK's list of english stopwords. Some examples include "i", "me", "myself", "we", and many more. These words are removed from the current text.

g. Lemmatization

Lemmatization is a form of word normalization, performed on each word to remove inflectional endings, obtaining the base form of a word, known as the lemma. For example:

```
dogs       -> dog
churches   -> church
aardwolves -> aardwolf
abaci      -> abacus
hardrock   -> hardrock
are        -> are
is         -> is
```

4. Cleaning

Entries with empty texts are removed from the data set. A total of 2383 (3.15%) emails were removed.

5. Saving

A `.csv` file with the 4 columns (index, is_spam, email_path, text) is stored as `processed.csv`, for later use, during model training. The first few lines of the file looks like this:

```
,is_spam,email_path,text
0,1,../data/inmail.1,feel pressure perform rising occasion try viagra
anxiety thing past back old self
1,0,../data/inmail.2,hi ive updated gulu check mirror seems little typo
debianreadme file example httpgulususherbrookecadebianreadme
ftpftpfrdebianorgdebianreadme testing lenny access release diststesting
```

```
current tested development snapshot named etch package tested unstable
passed automated test propogate release etch replace lenny like readmehtml
yan morin consultant en logiciel libre yanmorinsavoirfairelinuxcom
unsubscribe email debianmirrorsrequestlistsdebianorg subject unsubscribe
trouble contact listmasterlistsdebianorg
2,1,../data/inmail.3,mega authenticv g discount pricec l discount pricedo
miss click httpwwwmoujsjkhchumcom mailboundary authentic viagra mega
authenticv g discount pricec l discount pricedo miss click
3,1,../data/inmail.4,hey billy really fun going night talking said felt
insecure manhood noticed toilet quite small area worry website telling
secret weapon extra inch trust girl love bigger one ive time many chick
since used pill year ago package used month supply one worth every cent
website httpctmaycom ring weekend go drink let know secret later dude brad
```

6. Summary

	Original	**Cleaned**	**Remaining**
Total	75419	2383 (3%)	73036 (97%)
Spam	50199 (67%)	1854	24691
Ham	25220 (33%)	529	48345

7. Training and Testing Sets

The training and test sets are obtained at random when running train.py, using the `train_test_split` method of `scikit-learn`, a Python-based machine learning package. The method provides a parameter for the random seed called `random_state`, which allows for the same split, provided the same parameters. The `random_state` used was also 191.

The data set was split into the training set (80%) and the test set (20%), as such (with `random_state=191`):

	Train Set (80%)	**Test Set (20%)**
Total	58428	14608
Spam	38640 (66%)	9705 (66%)
Ham	19788 (34%)	4903 (34%)

Hyperparameter Optimization

In scikit-learn, there are multiple implementations for SVM, such as LinearSVC and SVC, there are also different parameters one could initialize these models as, such as the kernel, gamma, C, and degree (in the case of a polynomial kernel).

Thus, a grid search with three-fold cross validation was performed on 1000 rows, testing the following configurations:

```
params = [
    {
        'clf__estimator': [SVC()],
        'clf__estimator__C': [0.1, 1, 10, 100, 1000],
        'clf__estimator__kernel': ['rbf', 'linear', 'sigmoid'],
        'clf__estimator__gamma': ['scale', 0.1, 1, 10, 100]
    },
    {
        'clf__estimator': [LinearSVC()],
        'clf__estimator__C': [0.1, 1, 10, 100, 1000],
    },
    {
        'clf__estimator': [SVC()],
        'clf__estimator__C': [0.1, 1, 10, 100, 1000],
        'clf__estimator__kernel': ['poly'],
        'clf__estimator__degree': [2, 3, 4, 5, 6],
        'clf__estimator__gamma': ['scale', 0.1, 1, 10, 100]
    }
]
```

Two different classifiers are tested: **SVC**, and **LinearSVC**. The difference of LinearSVC is that it is "similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples."

This grid search results in **205 total combinations**, testing the various parameters and comparing their scores when trained and tested using three-fold cross-validation on a data set of 1000 (as testing these 205 combinations took so long when increasing the data set).

RESULTS

Training and Data Set

The data set was split into the training set (80%) and the test set (20%), as such (with `random_state=191`):

	Train Set (80%)	**Test Set (20%)**
Total	58428	14608
Spam	38640 (66%)	9705 (66%)

Ham	19788 (34%)	4903 (34%)

Hyperparameter Optimization

After performing grid search, the results were sorted based on mean scores, and the following are the results:

Rank	Model	Mean Score
1	SVC(kernel='sigmoid', C=10, gamma=1)	0.979994965
2	LinearSVC(C=100)	0.979991968
3	LinearSVC(C=10)	0.979991968
4	SVC(kernel='rbf', C=100, gamma='auto')	0.979991968
5	SVC(kernel='linear', C=10, gamma=10)	0.978990967
…	…	…
201	SVC(kernel='poly', degree=2, C=1, gamma=0.1)	0.749998501
202	SVC(kernel='poly', degree=2, C=1, gamma='scale')	0.749998501
203	SVC(kernel='sigmoid', C=10, gamma=100)	0.746006486
204	SVC(kernel='sigmoid', C=1000, gamma=100)	0.743992495
205	SVC(kernel='sigmoid', C=100, gamma=100)	0.735999473

Testing (Prediction & Evaluation)

Based on the results of grid search, we tested out the top 5 model configurations based on their mean scores, and here are the results after training them on the 58k training set, and testing the trained model on the remaining 14k testing set.

GS Rank	Model	Accuracy	Accu. Rank
1	SVC(kernel='sigmoid', C=10, gamma=1)	99.34967%	4
2	LinearSVC(C=100)	99.45920%	3
3	LinearSVC(C=10)	99.52081%	1
4	SVC(kernel='rbf', C=100, gamma='auto')	66.43620%	5
5	SVC(kernel='linear', C=10, gamma='10')	99.52081%	1

CONCLUSION

Two models scored the same accuracy, with almost the same parameters. The `LinearSVC(C=10)` and `SVC(kernel='linear', C=10, gamma='10')`. They're both linear implementations with the same C of 10, and both scored an accuracy of 99.52081%. Respectively, they also were rank 3 and 5 based on grid search results.

However, it is important to note that since the hyperparameter optimization using grid search only used 1000 of the data due to time constraints, there might have been a better model that could have scored higher than the two linear SVC models, but even so, an accuracy of 99.52081% provided the training and testing sets is significantly high enough.

REFERENCES

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., & Spyropoulos, C. D. (2000). An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*.

[2] Patel, S. (2017). Chapter 2 : SVM (Support Vector Machine) — Theory. Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72