TITLE

Alexander Björnsson
email1

Sara Sabrina Zemljič
sara.zemljic@gmail.com

School of Computer Science, Reykjavik University, Iceland

October 8, 2017

Abstract

We can all agree that the amount of spam we get every day by e-mail is almost limitless. The annoying spam messages are also getting more and more dangerous since they may contain viruses or other threats. Therefore it also ****(of?)**** no surprise that spam filters have been studied quite intesively with various methods from machine learning.

We present our models for spam filtering on the dataset spambase. ****ADD which models and how impressive they are =)****

1 Introduction

Following the definition in [2, p.2], **spam***: unwanted communication intended to be delivered to an indiscriminate target, directly or indirectly, notwithstanding measures to prevent its delivery;* **spam filter***: an automated technique to identify spam for the purpose of preventing its delivery.*.

– explain ham

– a word about false positives and why we want to minimize them

–

Let us just list a few models that have been trained for e-mail spam detection...

– a word about what has been studied on spam filters in general

– a word about what has been studied on (our) spambase and where we found it (!!!)

– maybe also why we decided for this dataset out of all others: the others would require way too much preprocesing with HTML and stuff like that which is not soooo relevant to this project

The rest of our paper is organized as follows. In the next section we introduce our dataset and what preprocesing we performed on it. In Section 3 we discuss the three classifiers we trained and their evaluation is presented in Section 4. We conclude the paper with final thoughts about the filters and present our code in Appendix.

2 Dataset

The dataset we are using for this research is the *spambase*, a SPAM E-mail Database [4] donated by George Forman (gforman at nospam hpl.hp.com, 650-857-7835) from Hewlett-Packard Labs and was generated in June/July 1999. The spam e-mails in the collection include advertisements for products or web sites, make money fast schemes, chain letters, etc. The collection of non-spam e-mails in the database came from filed work and personal e-mails, therefore the dataset

is very specific. For example words like 'george' or the code '650' are very strong indicators that an e-mail is not spam.

There are 4601 instances out of which 1813 (about 39.4%) are spam. Each instance is represented as a vector with 58 entries, so 57 + 1 columns, out of which

- the last one gives us the class information, it is either 1 (= spam) or 0 (= ham);
- first 48 columns are continuous real attributes in the range $[0, 100]$ of type `word_freq_WORD` (e.g. `word_freq_make`, `word_freq_address`, `word_freq_all`, etc.). These attributes present the ration of the number of times the `WORD` appears in the e-mail) over the total number of words in e-mail. A 'word' in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
- next 6 columns are continuous real attributes in the range $[0, 100]$ of type `char_freq_CHAR` (e.g. `char_freq_;`, `char_freq_!`, `char_freq_$`), which present the percentage of characters in the e-mail that match `CHAR`.
- last 3 attributes are continuous real in the range $[1, \ldots]$ and are a bit special. They count occurences with capital letters. For example, `capital_run_length_average` counts average length of *uninterrupted sequences of capital letters*.

This dataset is preprocesed, which means the attributes were chosen this way to classify spam the best. It also has no missing values. The data is not available in the raw format, so it is imposible to experiment with other attributes that could be extracted from their e-mails. Despite that this dataset was used in various studies for testing different classifiers for recognizing spam e-mails.

At the beginning we split our data into train and test sets, using 20% of the data for testing our models at the end. To ensure that we are always using the same train set we included the same `random_state` at all calculations. Our classes are in about $2 : 3$ ratio, so we checked that the train-test split has a similar ratio of spam and ham e-mails as well.

For some of our models we required normalized data, therefore we used **** which normalization?? ****

Preprocessing??

– Are we doing any preprocesing??

**** anything else missing?? ****

3 Models

We have decided to find a spam filter with our dataset with **** three **** different classifiers to determine which one gives us the best one. First we uses k nearest-neighbour classifier (kNN shortly), then we studied Naïve Bayes (NB) methods on our dataset and finally we built a neural network (NN) classifier.

**** maybe make a combined one from all three? ****

3.1 k nearest-neighbour classifier

– PCA cross validation (on standard k, which is 10??)

– – data is first split into train and test set, train set is then used for cross validation to determine which number of components would be the best to use; this is then used for (proper) PCA on the whole data set

–

– kNN is first trained on all 57 attributes and then compared with model obtained with "PCAed" data for the number of components that proved to be the best with cross-validation

– then kNN is hypertuned for which k would be the best

–

3.2 Bayesian ...

3.3 Neural networks

4 Evaluation of models

bla bla introduction text :D

4.1 kNN

– images for comparison

– confusion matrix

4.2 Bayesian ...

4.3 Neural networks

5 Conclusions

References

[1] I. Androutsopoulos, V. Metsis, G. Paliouras, Spam Filtering with Naive Bayes – Which Naive Bayes?, in: Proceedings of the Third Conference on Email and AntiSpam (2006).

[2] G. V. Cormack, Email Spam Filtering: A Systematic Review, Foundation and Trends in Information Retrieval (2006) 1(4) 335–455.

[3] I. Idris, E-mail Spam Classification With Artificial Neural Network and Negative Selection Algorithm, International Journal of Computer Science & Communication Networks 1(3) (2011) 227–231.

[4] M. Hopkins, E. Reeber, G. Forman, J. Suermondt, SPAM E-mail Database, Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304, `https://archive.ics.uci.edu/ml/datasets/spambase`.

[5] D. Puniškis, R. Laurutis, R. Dirmeikis, An artificial neural nets for spam email recognition, Elektronika ir Elektrotechnika (Electronics and Electrical Engineering) 5(69) (2006) 73–76.

[6] autors, title-of-article, journal issue-nr (year) pages–pages.

[7] autors, title-of-article, journal issue-nr (year) pages–pages.

[8] autors, title-of-article, journal issue-nr (year) pages–pages.

[9] autors, title-of-article, journal issue-nr (year) pages–pages.

[10] autors, title-of-article, journal issue-nr (year) pages–pages.

[11] autors, title-of-article, journal issue-nr (year) pages–pages.

Appendix: code

STRUCTURE THE CODE into smaller segments, we will only have main training parts here (no prints, no confusion matrices codes etc). Only main models and their preprocesing, for results we will just analyze them with words

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier
from  sklearn.model_selection import train_test_split

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from matplotlib.ticker import FuncFormatter
from sklearn.model_selection import KFold

from sklearn.metrics import confusion_matrix

"""
loading and adjusting data
"""

file = open("spambase.data")
data = np.loadtxt(file,delimiter=",")

X = data[:, 0:57]
y = data[:, 57]

dataframe = pd.DataFrame(data=X)

#apply normalization function to every attribute
dataframe_norm = dataframe.apply(lambda x: (x - np.mean(x)) / np.std(x))

X_train , X_test , y_train , y_test = train_test_split(dataframe_norm, y,
    test_size=0.20 , random_state=42)

"""
PCA
"""
def pca(n_components, dataframe):
    pca = PCA(n_components = n_components)
    pca.fit(dataframe)
    return pca.transform(dataframe)

"""
```

```
kNN model
"""


"""
NB model
"""


"""
ANN model
"""
```
