CS 314, Lab 6 - Report

Shriram Ghadge (180010015), Rishit Saiya (180010027)

March 26, 2021

1 Abstract

In this problem the task was to do Spam Email classification using Support Vector Machine. Using a SVM to classify emails into spam or non-spam categories and report the classification accuracy for various SVM parameters and kernel functions.

2 Dataset Description

An email is represented by various features like frequency of occurrences of certain keywords, length of capitalized words etc. A data set containing about 4601 instances are available in this source. The data format is also described in the above link. You have to randomly pick 70% of the data set as training data and the remaining as test data.

3 Libraries & Packages

The below are the libraries and packages used in the code:

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
```

3.1 Pandas Package

`pandas` is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal.

3.2 Scikit-Learn Package

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4 Kernels

The following subsections explain the kernels used in the code to implement SVM for the given dataset.

4.1 Linear Kernel

Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set.

4.2 Polynomial Kernel

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.

Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In the context of regression analysis, such combinations are known as interaction features. The (implicit) feature space of a polynomial kernel is equivalent to that of polynomial regression, but without the combinatorial blowup in the number of parameters to be learned. When the input features are binary-valued (booleans), then the features correspond to logical conjunctions of input features.

In this problem, the degree is assumed as $\boldsymbol{2}$.

4.3 RBF Kernel

In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

5 Result & Analysis

The Figure 1 shows the accuracy obtained across different C parameters and different kernels.

	C	Linear	Poly	RBF
0	0.00005	(0.615527950310559, 0.6017378711078928)	(0.6105590062111801, 0.5959449674149168)	(0.610248447204969, 0.5959449674149168)
1	0.00050	(0.8664596273291926, 0.8464880521361332)	(0.6142857142857143, 0.5981173062997828)	(0.610248447204969, 0.5959449674149168)
2	0.00500	(0.922360248447205, 0.9029688631426502)	(0.6226708074534162, 0.6053584359160029)	(0.610248447204969, 0.5959449674149168)
3	0.05000	(0.9360248447204969, 0.9181752353367125)	(0.6822981366459627, 0.662563359884142)	(0.8981366459627329, 0.887762490948588)
4	0.50000	(0.9403726708074535, 0.9160028964518465)	(0.7614906832298136, 0.7313540912382331)	(0.9425465838509317, 0.9174511223750905)
5	1.00000	(0.9425465838509317, 0.9145546705286025)	(0.7925465838509317, 0.7552498189717596)	(0.9503105590062112, 0.9261404779145547)
6	5.00000	(0.9434782608695652, 0.9160028964518465)	(0.8673913043478261, 0.8269370021723389)	(0.9645962732919254, 0.9326574945691528)
7	50.00000	(0.94472049689441, 0.9167270094134685)	(0.9521739130434783, 0.905141202027516 2)	(0.9832298136645963, 0.9261404779145547)
8	500.00000	(0.9450310559006211, 0.9160028964518465)	(0.9829192546583851, 0.9203475742215785)	(0.9922360248447205, 0.9232440260680667)
9	5000.00000	(0.7006211180124223, 0.6886314265025344)	(0.9937888198757764, 0.9174511223750905)	(0.996583850931677, 0.9044170890658942)
10	50000.00000	(0.9211180124223602, 0.8971759594496741)	(0.9962732919254659, 0.9036929761042722)	(0.9972049689440994, 0.9044170890658942)

Figure 1: Accuracy obtained across different C parameters and different kernels

For a given C and a given kernel, the pair `(a,b)` is obtained. `a` is the training accuracy and `b` is the testing accuracy.

In the above mentioned dataset, for `C = 500`, we obtain the highest accuracy for testing & training dataset in the linear kernel mode implementation. Also, for $C \geq 500$ onwards we got maximum accuracy for the quadratic kernel. At the last, for $C \geq 500$, onwards we got maximum accuracy for training & testing accuracy in the RBF kernel.

6 Conclusion

The training data is linear separable because for kernel other than linear model perform poorly with low value of C. For very tiny values of C, we should get mis classified examples, often even if our training data is linearly separable. So, this is the final conclusion across different C parameter and various kernels for the given dataset.