CS 312, Lab 6 - Report

Niteesh Kamal Chaudhary (200010035)

March 16, 2022

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.

SVM

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane.

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 is given. The data format is also described in the above link. You have to randomly pick 80% of the data set as training data and the remaining as test data.

Libraries & Packages

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

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
```

Pandas Package

Pandas are 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.

```
data = pd.read_csv('spambase.data', sep=',', header=None)
data.head()
data.tail()
```

Standard Scaler

Used for preprocessing of data by transforming it in to standard scaler.

```
x_train = StandardScaler().fit_transform(x_train)
x_test = StandardScaler().fit_transform(x_test)
```

$$z = (x - u) / s$$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False.

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.

```
C_values = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100,500]  #hyper parameters
kernals = ['linear', 'poly', 'rbf'] #kernels
```

C parameter

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly

Kernels

A kernel is a function used in SVM for helping to solve problems. Using kernel, we can go to higher dimensions and perform smooth calculations with the help of it. We can go up to an infinite number of dimensions. The following subsections explain the kernels used in the code to implement SVM for the
given dataset.

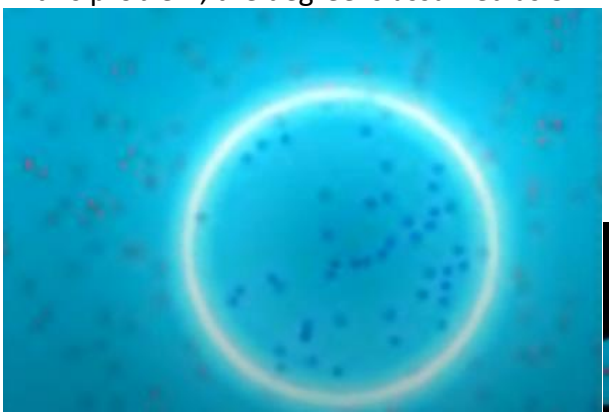
1) Linear Kernel

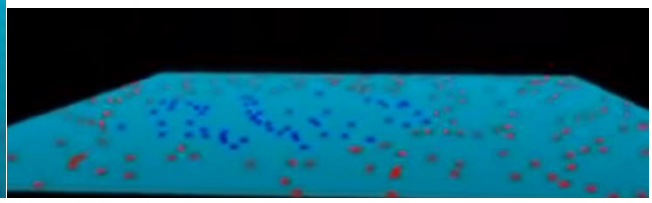
Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. The equation of the hyperplane in the 'M' dimension can be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

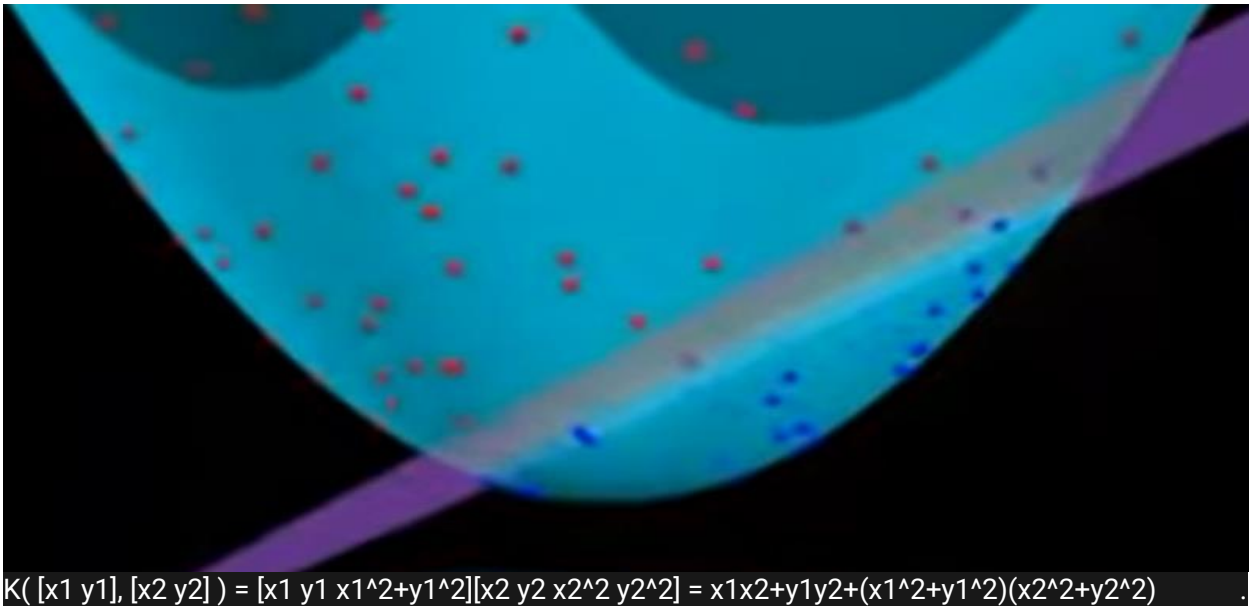
$$\begin{aligned} y &= w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 \ldots . \\ &= w_0 + \sum_{i=1}^{m} w_i x_i \\ &= w_0 + w^T X \\ &= b + w^T X \end{aligned}$$

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 combifactorial blow up 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. The kernel is the dot product in the higher-dimensional space
In this problem, the degree is assumed as 3.







K([x1 y1], [x2 y2]) = [x1 y1 x1^2+y1^2][x2 y2 x2^2 y2^2] = x1x2+y1y2+(x1^2+y1^2)(x2^2+y2^2) .

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 like non-linear datasets and you can't seem to figure out the right feature transform or the right kernel to use. RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

Result & Analysis

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

C	Linear	Poly	RBF
0.0001	(0.7361413043478261, 0.7643865363735071)	(0.6067934782608696, 0.6112920738327905)	(0.6046195652173914, 0.6112920738327905)
0.001	(0.8896739130434783, 0.9055374592833876)	(0.6095108695652174, 0.6134636264929425)	(0.6046195652173914, 0.6112920738327905)
0.01	(0.9192934782608696, 0.9261672095548317)	(0.6307065217391304, 0.6503800217155266)	(0.752445652173913, 0.743756786102063)
0.1	(0.9296195652173913, 0.9326818675352877)	(0.7002717391304348, 0.7198697068403909)	(0.9144021739130435, 0.9055374592833876)
1	(0.9317934782608696, 0.9337676438653637)	(0.7991847826086956, 0.7926167209554832)	(0.9451086956521739, 0.9359391965255157)
5	(0.9315217391304348, 0.9348534201954397)	(0.8679347826086956, 0.8577633007600435)	(0.9595108695652174, 0.9359391965255157)
10	(0.9339673913043478, 0.9305103148751357)	(0.8991847826086956, 0.8838219326818675)	(0.9644021739130435, 0.9326818675352877)
100	(0.9326086956521739, 0.9250814332247557)	(0.9627717391304348, 0.9337676438653637)	(0.9855978260869566, 0.9272529858849077)
300	(0.9334239130434783, 0.9250814332247557)	(0.9769021739130435, 0.9348534201954397)	(0.991304347826087, 0.9185667752442996)
500	(0.9328804347826087, 0.9239956568946797)	(0.9828804347826087, 0.9370249728555917)	(0.9921195652173913, 0.9218241042345277)

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 = 100, we obtain the highest accuracy for testing & Training dataset in the linear kernel mode implementation. Also, for Cv 100 onwards we got maximum accuracy for the quadratic kernel. At the last, for Cv 100, onwards we got maximum accuracy for training & testing accuracy in the RBF kernel.

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.