****

**(B.TECH) Trimester-XI AY 2020-21**

**PPL Lab Assignment No. 04**

**Faculty: Prof. Jayshree Aher**

**Problem Statement:** To study/ implement SVM.

**Name: Vishwa Mohan Singh**

**ERP No.: 1032170273**

**Panel: A (AML2)**

**Date: 4/1/2021**

**Objectives:**

1. To understand the basics of SVM.
2. To analyze the classifier with kernel functions.

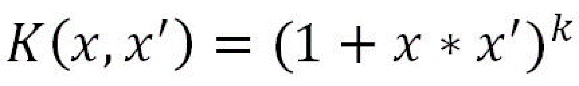
**Theory:** (describe the following)

* **Support Vector Machine**: Support Vector Machines are supervised learning models mainly used for classification and regression, that uses the edge points from the dataset called support vectors, to estimate the classification/regression line which in this case is called a hyper-plane.
* **Different Kernel Functions:** Sometimes in a lower dimensional space, it is not possible to draw the hyperplane in an ideal way. Therefore, we sometimes need to expand the dimension in a peculiar manner in order to spread the dataset such that the hyperplane line obtains the minimum error. This dimension is increased using a function called the kernel.

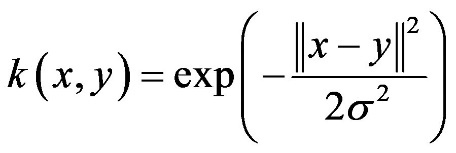
**Some important kernels**

**Linear:** This is the most basic kernel type where the dimension remains unchanged.

**Polynomial kernel**: This works by creating a polynomial equation from the dataset, which determines the third dimension.



**Gaussian Radial Basis Function:** This is a similarity function that uses a centroid and scores every point between 0 to 1 depending on how close the point is to the centroid, factored in by the parzen’s window’s value.



* Advantages of SVM
  + Works relatively better when there is a clear margin of saperation.
  + More effective in high dimensional spaces.
  + Relatively, memory efficient.
  + Effective in cases where the number of dimensions is greater than the number of samples.

**Operations to be performed on dataset:**

**Steps in Preprocessing of Data**

1. Read the .csv file of dataset
2. Display few observations
3. Perform data preprocessing (handling missing data, etc)
4. Create the independent and dependent variables
5. Standardization of data
6. Reduce the data dimension
7. Split the data into training and test sets.
8. Create the object of classifier with linear and rbf kernel.
9. Fit the data in model to train it.
10. Try to plot the decision boundary for training set and testing set.

**Program code:**

import pandas as pd

import numpy as np

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from sklearn.decomposition import PCA

## Importing the data

dataset = pd.read\_csv("train.csv")

## Array Transformation

arr = dataset.iloc[:,[2,5,6,7,9,11,1]]

## Handeling NaN

impu = SimpleImputer(missing\_values=np.nan, strategy = "mean")

arr = arr.values

arr[:,[0,1,2,3,4]] = impu.fit\_transform(arr[:,[0,1,2,3,4]])

#Remvoing the entries from categorical columns

arr = pd.DataFrame(arr)

arr = arr.dropna()

arr = arr.values

#OneHot and LableEncoding

lb = LabelEncoder()

arr[:,-1] = lb.fit\_transform(arr[:,-1])

transformer = ColumnTransformer(

transformers=[("OneHot", OneHotEncoder(),[5])],

remainder='passthrough'

)

arr = transformer.fit\_transform(arr.tolist())

##X-Y Split

X = arr[:,:-1]

Y = arr[:,-1].astype('int')

#Scaling

scaler = StandardScaler()

X = scaler.fit\_transform(X)

#Reducing dimensions to 2

pca = PCA(n\_components=2)

X = pca.fit\_transform(X)

#Train Test Split

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.2, random\_state=10)

##Importing the libraries for creating the models

from sklearn.svm import SVC # For support vector machines

#SVM with linear kernel

model= SVC(kernel = 'linear', probability=True)

print("Training Support Vector Machine with linear kernel")

model.fit(X\_train,Y\_train)

#Accuracy

print("Accuracy: ", model.score(X\_test,Y\_test))

#Confusion matrix

print("Confusion Matrix: \n", confusion\_matrix(Y\_test,model.predict(X\_test)))

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, Y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM Linear')

plt.xlabel('Reduced Feature 1')

plt.ylabel('Reduced Feature 2')

plt.legend()

plt.show()

#Breaking

print()

#SVM with rbf kernel

model= SVC(kernel = 'rbf', probability=True)

print("Training Support Vector Machine with RBF Kernel")

model.fit(X\_train,Y\_train)

#Accuracy

print("Accuracy: ", model.score(X\_test,Y\_test))

#Confusion matrix

print("Confusion Matrix: \n", confusion\_matrix(Y\_test,model.predict(X\_test)))

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, Y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM RBF')

plt.xlabel('Reduced Feature 1')

plt.ylabel('Reduced Feature 2')

plt.legend()

plt.show()

#Breaking

print()

#SVM with polynomial kernel

model= SVC(kernel = 'poly', probability=True)

print("Training Support Vector Machine with Polynomial Kernel")

model.fit(X\_train,Y\_train)

#Accuracy

print("Accuracy: ", model.score(X\_test,Y\_test))

#Confusion matrix

print("Confusion Matrix: \n", confusion\_matrix(Y\_test,model.predict(X\_test)))

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, Y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, model.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('SVM POLY')

plt.xlabel('Reduced Feature 1')

plt.ylabel('Reduced Feature 2')

plt.legend()

plt.show()

#Breaking

print()

**Dataset used: (source link & description)**

**Titanic:** A Kaggle competition dataset that is used for a machine learning competition aimed at predicting who might have survived the sinking. It includes passenger information like class, embarkment port, fare, cabin etc.

**Link:** <https://www.kaggle.com/c/titanic/data>

**Output:**

Training Support Vector Machine with linear kernel

Accuracy: 0.7303370786516854

Confusion Matrix:

[[104 11]

[ 37 26]]

Training Support Vector Machine with RBF Kernel

Accuracy: 0.7584269662921348

Confusion Matrix:

[[103 12]

[ 31 32]]

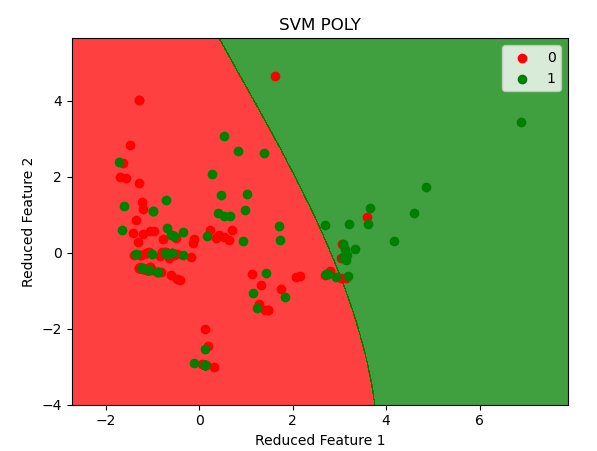
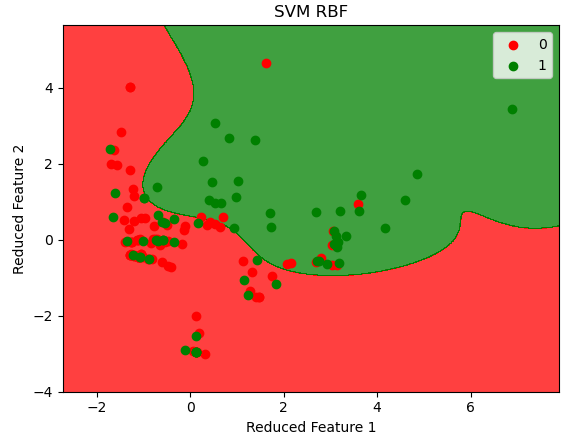
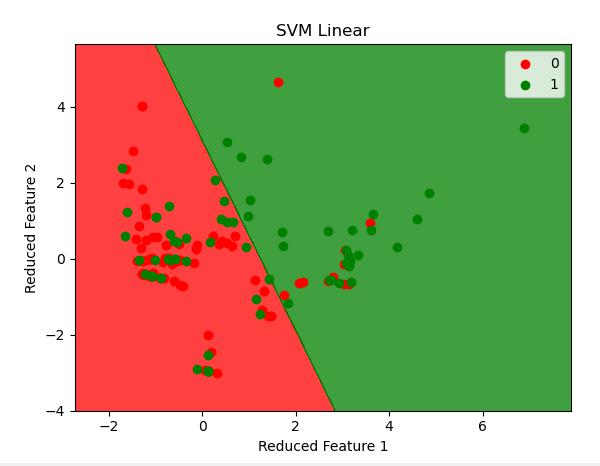
Training Support Vector Machine with Polynomial Kernel

Accuracy: 0.702247191011236

Confusion Matrix:

[[110 5]

[ 48 15]]



**FAQs:**

1. **What is a Classifier?**

Classifier is a model that is used to predict categorical or binary output. When the dependent variable is finite and not continuous, spread over a long range, we predict the value using classifiers.

1. **Compare SVM with at least 03 other classifiers**.

|  |  |  |  |
| --- | --- | --- | --- |
| **SVM** | **Logistic/Linear** | **Decision Tree** | **Neural Network** |
| Uses hyperplanes based on support vectors to classify | Uses a line similar to a hyperplane | Uses trees to classify | Uses perceptron layers to classify |
| Gets the plane based on the support vector | Gets the line based on the OLS calculations | Gets the ideal tree by reducing the entropy | Uses backpropagation to get the ideal weights |
| Uses various types of kernels to improve performance | Only compatible with polynomial kernels | No kernel is required to make a tree | Uses kernels as activation function at each perceptron. |
|  |  |  |  |

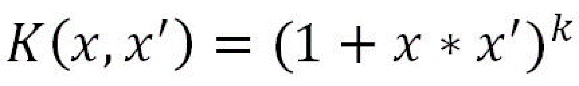
1. **How would you tune SVM parameters?**

The main parameters in SVM are the kernel, the penalty and the gamma value. To tune this, we have to try multiple permutations of these parameters to get the ideal combination. The easiest way to perform this is by using a method called **grid searching**: where we can pass a list of possible values for the parameters and using an evaluation criterion, the grid search algorithm will find the ideal combination from the given options. For the evaluation in grid searching, the ideal method is called the **k-fold cross validation**, where the test set is divided into k segments of which, k-1 segments are given for training and a random remaining segment is kept for testing. This is done multiple times until every segment has been on the test side. All the calculated accuracies are aggregated, which becomes the final result.

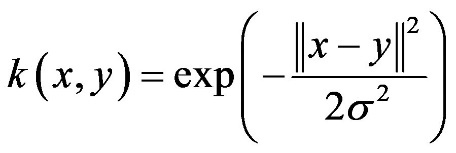
1. **Describe various types of kernel functions.**

**Linear:** This is the most basic kernel type where the dimension remains unchanged.

**Polynomial kernel**: This works by creating a polynomial equation from the dataset, which determines the third dimension.



**Gaussian Radial Basis Function:** This is a similarity function that uses a centroid and scores every point between 0 to 1 depending on how close the point is to the centroid, factored in by the parzen’s window’s value.



1. **Give the applications of SVM classifier.**
   1. Used in applications like facial recognition.
   2. The suitability to a high-dimensional set makes it appropriate for BCI applications.
   3. Used in bioinformatic problems like protein folding and report diagnosis.
   4. In various computer vision tasks like handwriting recognition and image classification.
2. **State the significance of the kernel function.**

Sometimes in a lower dimensional space, it is not possible to draw the hyperplane in an ideal way. Therefore, we sometimes need to expand the dimension in a peculiar manner in order to spread the dataset such that the hyperplane line obtains the minimum error. This dimension is increased using a function called the kernel.

**Conclusion:**

SVM classifier was studied and the implementation was performed for the kernel functions.