 **GHARDA INSTITUTE OF TECHNOLOGY** 

***Department of Computer Engineering***

**Machine Learning Lab BE Computer (Semester-VII)**

**Experiment No.5 : Support Vector Machines (SVM)**

**Aim**- To study, understand and implement a SVM algorithm.

**Theory**-

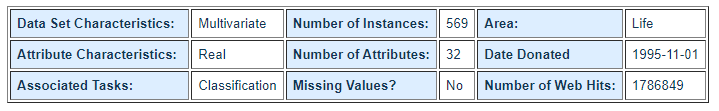
In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

**What does SVM do?**

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

**\*\* About Dataset-** Title: Breast Cancer Wisconsin (Diagnostic) Data Set



Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) , a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is described in [3].

**Attribute Information:**

1) ID number

2) Diagnosis / Classes (M = malignant, B = benign)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

Below is attribute information with their range:

# Attribute Domain / Range

-- -----------------------------------------

1. Sample code number id number

2. Clump Thickness 1 - 10

3. Uniformity of Cell Size 1 - 10

4. Uniformity of Cell Shape 1 - 10

5. Marginal Adhesion 1 - 10

6. Single Epithelial Cell Size 1 - 10

7. Bare Nuclei 1 - 10

8. Bland Chromatin 1 - 10

9. Normal Nucleoli 1 - 10

10. Mitoses 1 - 10

11. Class: (2 for benign, 4 for malignant)

**Code -**

from google.colab import files

up=files.upload()

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# reading csv file and extracting class column to y.

x = pd.read\_csv("breast-cancer-wisconsin.data")

print(x.dtypes)

x.columns =['Sample code number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape','Marginal Adhesion','Single Epithelial Cell Size','Bare Nuclei','Bland Chromatin','Normal Nucleoli','Mitoses','Class']

print(x)

x.drop(['Bare Nuclei'], axis=1, inplace=True)

print(x.dtypes)

x.shape

a = np.array(x)

print(a)

a.shape

y = a[:,9] # classes having 2 and 4

x=a[:,1:9]

print(x)

from sklearn.svm import SVC

clf = SVC(kernel='linear')

# fitting x samples and y classes

clf.fit(x, y)

clf.predict([[5, 9,6,3,4,2,5,2]])

Change the kernel type to rbf in below line and look at the impact

from sklearn import svm

svc = svm.SVC(kernel='rbf', C=1,gamma='auto').fit(x, y)

svc.predict([[5, 9,6,3,4,2,5,2]])

**Results & Discussion-**

## **Pros and Cons**-

* Pros:
  + It works really well with a clear margin of separation
  + It is effective in high dimensional spaces and in cases where the number of dimensions is greater than the number of samples.
  + It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
* Cons:
  + It doesn’t perform well when we have large data set (as the required training time is higher) and when the data set has more noise i.e. target classes are overlapping
  + SVM doesn’t directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

**References-**

1. <http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>
2. K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992
3. K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34