National University of Computer & Emerging Sciences, Karachi

FALL 2024, LAB MANUAL – 07

SUPPORT VECTOR MACHINE

COURSE CODE :	**AL3002**
INSTRUCTOR :	**Usama Bin Umar**

OBJECTIVE

1. Understanding of SVM
2. How SVM Works for classification
3. Support Vector Kernels
4. Implementation of SVC

SUPPORT VECTOR MACHINE

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

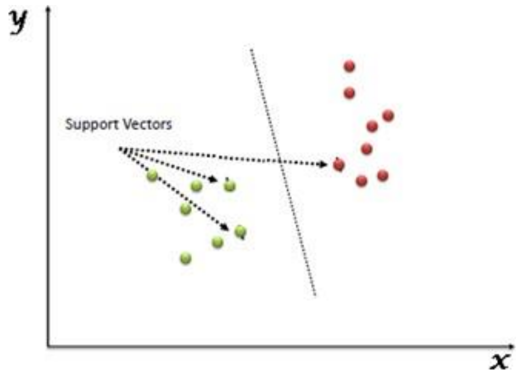
Before diving into the working of SVM let's first understand the two basic terms used in the algorithm "The support vector" and "Hyper-Plane".

HYPER-PLANE

A hyperplane is a decision boundary that differentiates the two classes in SVM. A data point falling on either side of the hyperplane can be attributed to different classes. The dimension of the hyperplane depends on the number of input features in the dataset. If we have 2 input features the hyper-plane will be a line. Likewise, if the number of features is 3, it will become a two-dimensional plane.

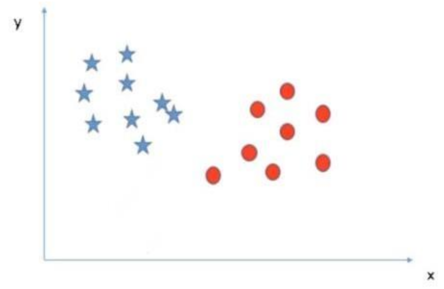
SUPPORT-VECTORS

Support vectors are the data points that are nearest to the hyper-plane and affect the position and orientation of the hyper-plane. We have to select a hyperplane, for which the margin, i.e the distance between support vectors and hyper-plane is maximum. Even a little interference in the position of these support vectors can change the hyper-plane.

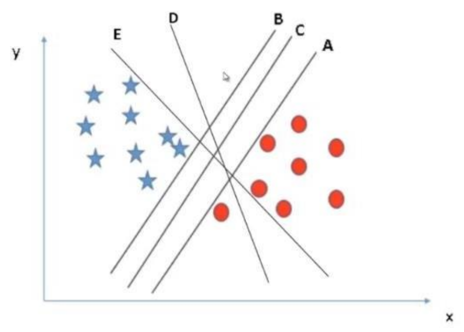


WORKING OF SVM

Let’s take an example, we have a classification problem where we have to separate the red data points from the blue ones.



Since it is a two-dimensional problem, our decision boundary will be a line, for the 3-dimensional problem we have to use a plane, and similarly, the complexity of the solution will increase with the rising number of features.

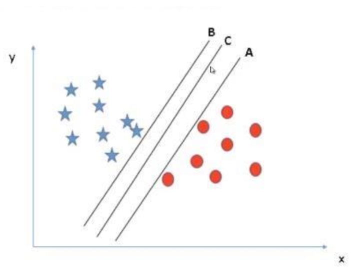


As shown in the above image, we have multiple lines separating the data points successfully. But our objective is to look for the best solution.

There are few rules that can help us to identify the best line.

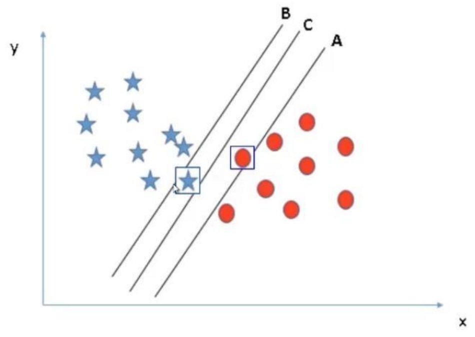
- Maximum Classification
- Best Separation

Maximum classification, i.e the selected line must be able to successfully segregate all the data points into the respective classes. In our example, we can clearly see lines E and D are miss classifying a red data point. Hence, for this problem lines A, B, C is better than E and D. So we will drop them.



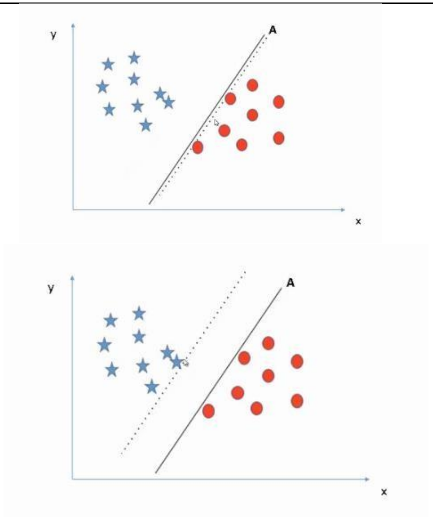
The second rule is **Best Separation**, which means, we must choose a line such that it is perfectly able to separate the points.

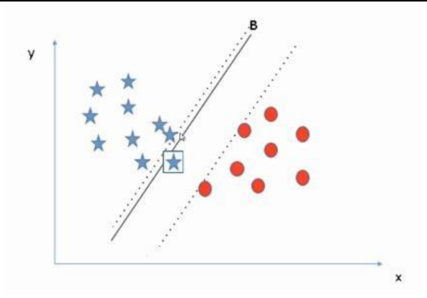
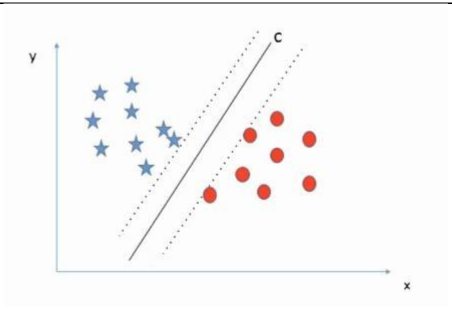
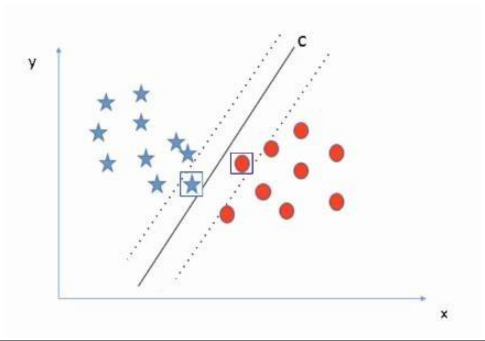
If we talk about our example, if we get a new red data point closer to line A as shown in the image below, line A will miss classifying that point. Similarly, if we got a new blue instance closer to line B, then line A and C will classify the data successfully, whereas line B will miss classifying this new point.



The point to be noticed here, In both the cases line C is successfully classifying all the data points why? To understand this let’s take all the lines one by one.

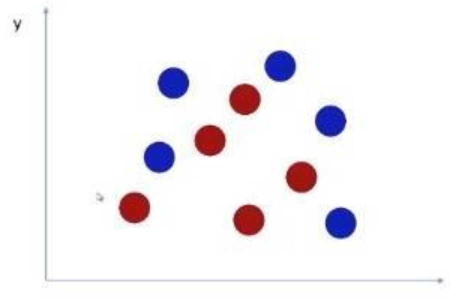
Let’s Discuss Which Line to Choose

Why not Line A	
First, consider line A. If I move line A towards the left, we can see it has very little chance to miss classify the blue points. on the other hand, if I shift line A towards the right side it will very easily miss-classify the red points. The reason is on the left side of the margin i.e the distance between the nearest data point and the line is large whereas on the right side the margin is very low.	

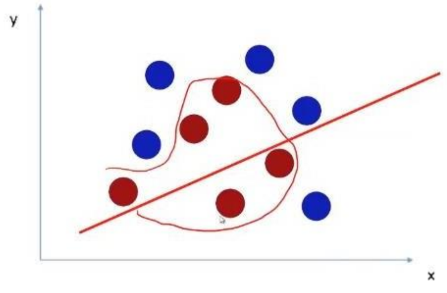
Why not Line B	
Similarly, in the case of line B. If we shift line B towards the right side, it has a sufficient margin on the right side whereas it will wrongly classify the instances on the left side as the margin towards the left is very low. Hence, B is not our perfect solution.	
Why not Line C	
In the case of line C, It has sufficient margin on the left as well as the right side. This maximum margin makes line C more robust for the new data points that might appear in the future. Hence, C is the best fit in that case that successfully classifies all the data points with the maximum margin.	 

This is what SVM looks for, it aims for the maximum margin and creates a line that is equidistant from both sides, which is line C in our case. so we can say C represents the SVM classifier with the maximum margin.

Now let’s look at the data below, As we can see this is not linearly separable data, so SVM will not work in this situation. If anyhow we try to classify this data with a line, the result will not be promising.

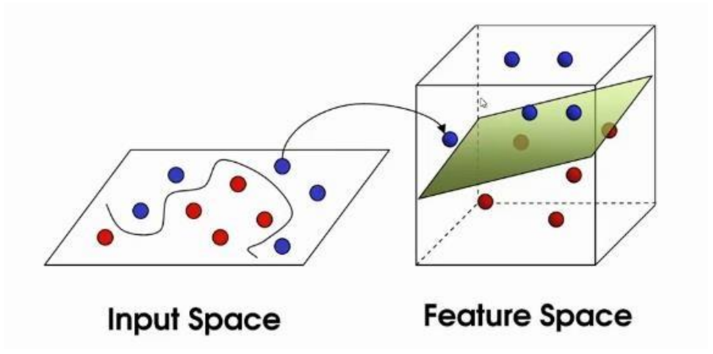


So, is there any way that SVM can classify this kind of data? For this problem, we have to create a decision boundary that looks something like this.



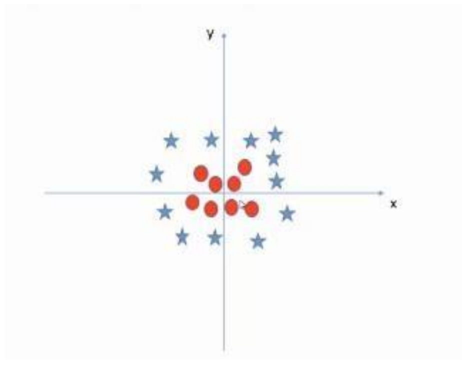
The question is, is it possible to create such a decision boundary using SVM. Well, the answer is **Yes**. SVM does this by projecting the data in a higher dimension. As shown in the following image. In the first case, data is not linearly separable, hence, we project into a higher dimension.

If we have more complex data then SVM will continue to project the data in a higher dimension till it becomes linearly separable. Once the data become linearly separable, we can use SVM to classify just like the previous problems.



Projection into Higher Dimension

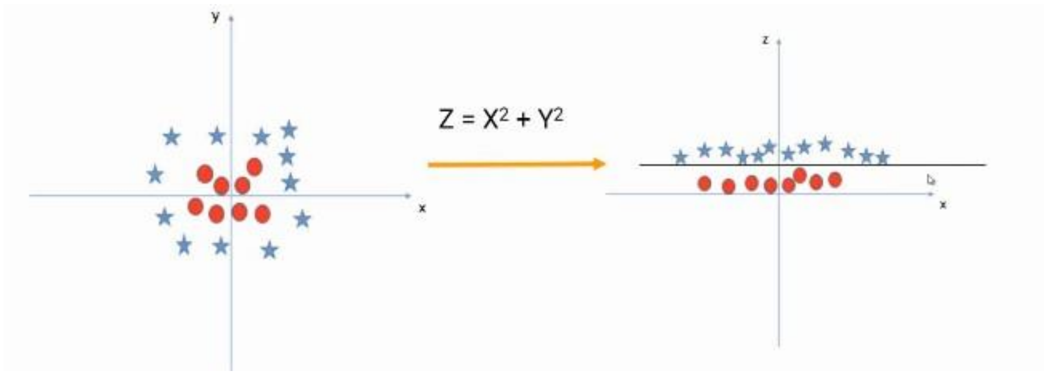
Now let’s understand how SVM projects the data into a higher dimension. Take this data, it is a circular non linearly separable dataset.



To project the data in a higher dimension, we are going to create another dimension z, where

$$Z = X^2 + Y^2$$

Now we will plot this feature Z with respect to x, which will give us linearly separable data that looks like this



Here, we have created a mapping Z using the base features X and Y, this process is known as **kernel transformation**. Precisely, a kernel takes the features as input and creates the linearly separable data in a higher dimension.

Now the question is, do we have to perform this transformation manually? The answer is no. SVM handles this process itself, just we have to choose the kernel type.

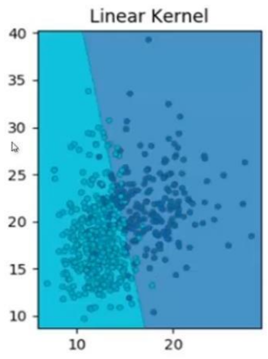
Implementation of SVC

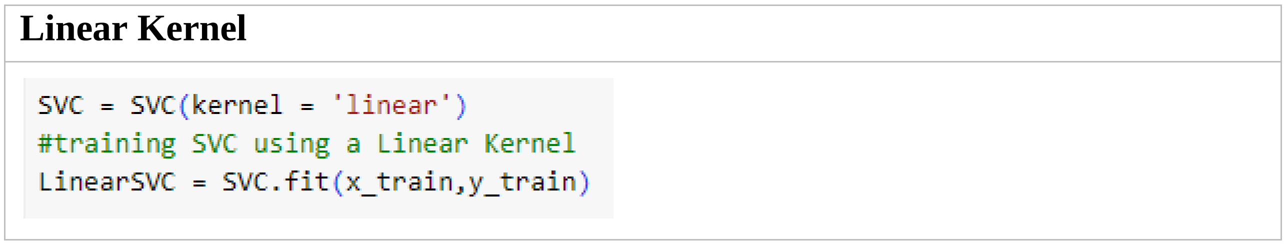
```python
from sklearn.svm import SVC
SVC = SVC()
modelSVC = SVC.fit(x_train,y_train)
prediction= modelSVC.predict(x_test)

from sklearn.metrics import accuracy_score
print("=====================Training Accuarcy=============")
trac=SVC.score(x_train,y_train)
print(trac)
print("=====================Testing Accuarcy=============")
teacSVC=accuracy_score(y_test,prediction)
print(teacSVC)
```

TYPES OF KERNEL

Linear Kernel

To start with, in the linear kernel, the decision boundary is a straight line. Unfortunately, most of the real-world data is not linearly separable, this is the reason the linear kernel is not widely used in SVM.



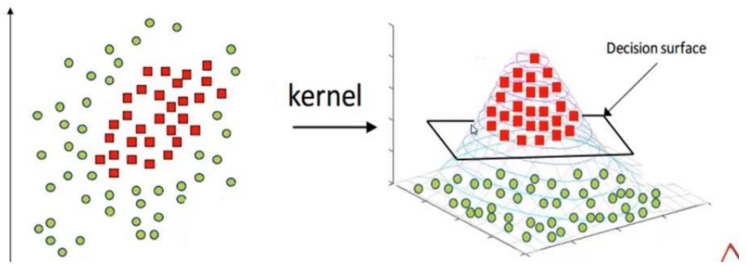


Linear Kernel

```
SVC = SVC(kernel = 'linear')
#training SVC using a Linear Kernel
LinearSVC = SVC.fit(x_train,y_train)
```

Gaussian / RBF kernel

It is the most commonly used kernel. It projects the data into a Gaussian distribution, where the red points become the peak of the Gaussian surface and the green data points become the base of the surface, making the data linearly separable.

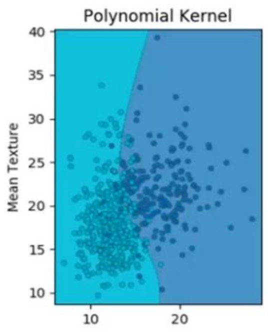


But this kernel is prone to overfitting and it captures the noise.

Gaussian Kernel

```
SVC = SVC(kernel = 'rbf')
#training SVC using a Gaussian (rbf) Kernel
RbfSVC = SVC.fit(x_train,y_train)
```

Polynomial kernel
At last, we have a polynomial kernel, which is non-uniform in nature due to the polynomial combination of the base features. It often gives good results.



But the problem with the polynomial kernel is, the number of higher dimension features increases exponentially. As a result, this is computationally more expensive than RBF or linear kernel.

Polynomial Kernel

```
SVC = SVC(kernel = 'poly' ,degree = 4)
#training SVC using a Polynomial Kernel
polynomialSVC = SVC.fit(x_train,y_train)
```

Hard margin
In a hard margin SVM, the goal is to find the hyperplane that can perfectly separate the data into two classes without any misclassification. However, this is not always possible when the data is not linearly separable or contains outliers. In such cases, the hard margin SVM will fail to find a hyperplane that can perfectly separate the data, and the optimization problem will have no solution.

Soft Margin

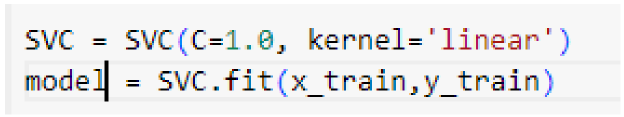
In a soft margin SVM, we allow some misclassification by introducing slack variables that allow some data points to be on the wrong side of the margin. The optimization problem in a soft margin SVM is modified as follows:

$$\textbf{minimize } 1/2\ ||w||^2 + C * \Sigma\xi i$$

$$\textbf{subject to } yi(w^\wedge T\ xi + b) \geq 1 - \xi i$$

=> where ξi are the slack variables, and C is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the misclassification. A larger value of C results in a narrow margin and fewer misclassifications, while a smaller value of C results in a wider margin but more misclassifications.
Geometrically, the soft margin SVM introduces a penalty for the data points that lie on the wrong side of the margin or even on the wrong side of the hyperplane. The slack variables ξi allow these data points to be within the margin or on the wrong side of the hyperplane, but they incur a penalty in the objective function. The optimization problem in a soft margin SVM finds the hyperplane that maximizes the margin while minimizing the penalty for the misclassified data points.

C Parameter



```
SVC = SVC(C=1.0, kernel='linear')
model = SVC.fit(x_train,y_train)
```

Gamma Parameter
It tells us how much will be the influence of the individual data points on the decision boundary.

– Large Gamma: Fewer data points will influence the decision boundary. Therefore, decision boundary becomes non-linear leading to overfitting

– Small Gamma: More data points will influence the decision boundary. Therefore, the decision boundary is more generic.

Gamma Parameter

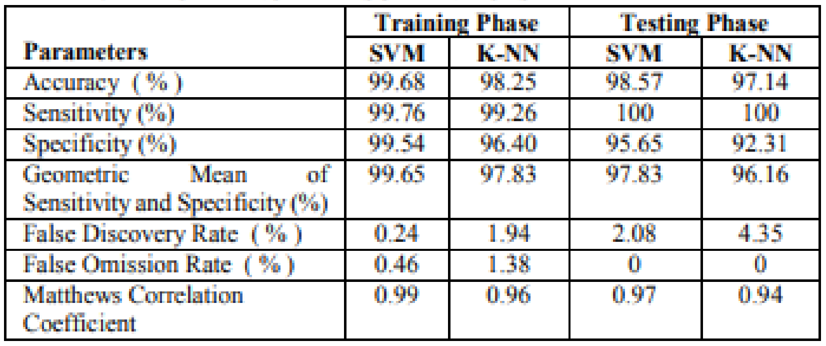
TASKS

TASK 1:

Download the dataset

- Perform EDA and all necessary analysis as performed by the author of this article
- Apply KNN and SVM
- Apply Evaluation Metrics that has been used in this paper (See Table 1)

TABLE I. PERFORMANCE MEASURE INDICES



Parameters	Training Phase		Testing Phase	
	SVM	K-NN	SVM	K-NN
Accuracy (%)	99.68	98.25	98.57	97.14
Sensitivity (%)	99.76	99.26	100	100
Specificity (%)	99.54	96.40	95.65	92.31
Geometric Mean of Sensitivity and Specificity (%)	99.65	97.83	97.83	96.16
False Discovery Rate (%)	0.24	1.94	2.08	4.35
False Omission Rate (%)	0.46	1.38	0	0
Matthews Correlation Coefficient	0.99	0.96	0.97	0.94

- Generate the result of Figure 1 and 2 (See Paper)

TASK 2:

Work on the same dataset , and now use this paper as a reference

- Perform all necessary analysis as performed by the author of this article
- Do perform feature selection with the same technique as mentioned by the author
- Generate the results of having various categories of data division as mentioned in Table4
- Perform feature selection on every subset as obtained in Table 2
- Genrate Table 3 and Check whether you are obtaining the same features on the same division.
- Generate table 6 and 7

TASK 3:

- Work on the same dataset , with complete EDA and Data Wrangling
- Apply SVM with all the Kernel
- Adjust the SVM Hyper parameter Gamma and C and compare your results with default parameters