Support Vector Machines
(Non-Linear)

Presented By:

Divyash Kumbhani

Jay Kanani

Sagar Bapodara

Instructor: Dr. Yaseer Alginahi

Date: October 20th, 2023

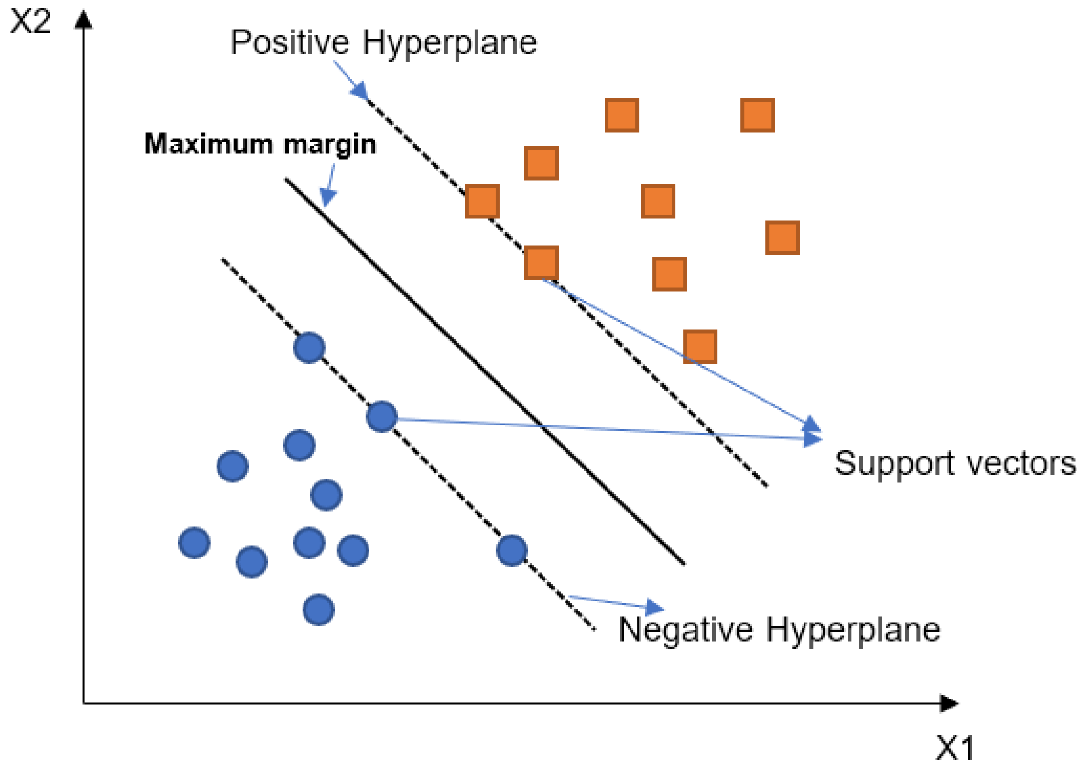
Table of Contents

- **Introduction**
  - Types of SVM
  - Important Terminologies
  - Working of Non-Linear SVMs
- **Mathematics**
  - Mathematics Behind Kernels
  - Numerical Example
  - Evaluation Metrics
- **Implementation**
  - Applications
  - Example
  - Advantage and Limitations
  - Overcoming Limitations

Introduction

- Support Vector Machine (SVM) for Classification & Regression
- Hyperplane to separate the classes.
- Support vectors and margin for hyperplane.

Decision boundary and hyperplane for SVM.

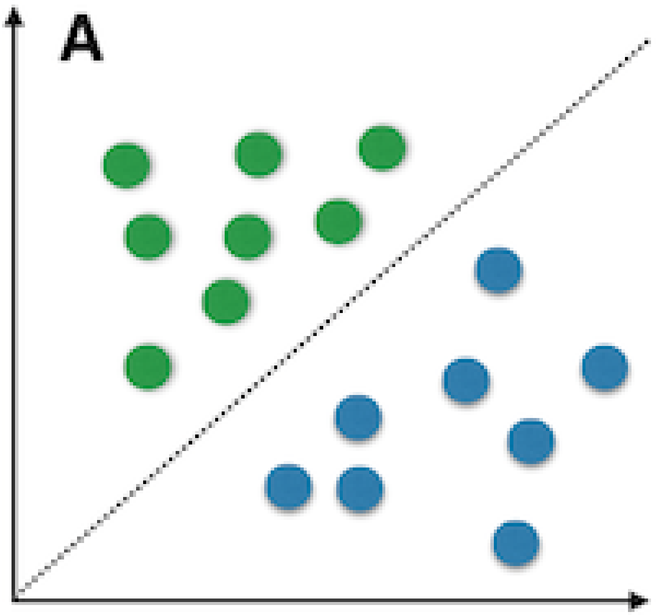


Source: https://tinyurl.com/3r8cbjdk

Types of SVM

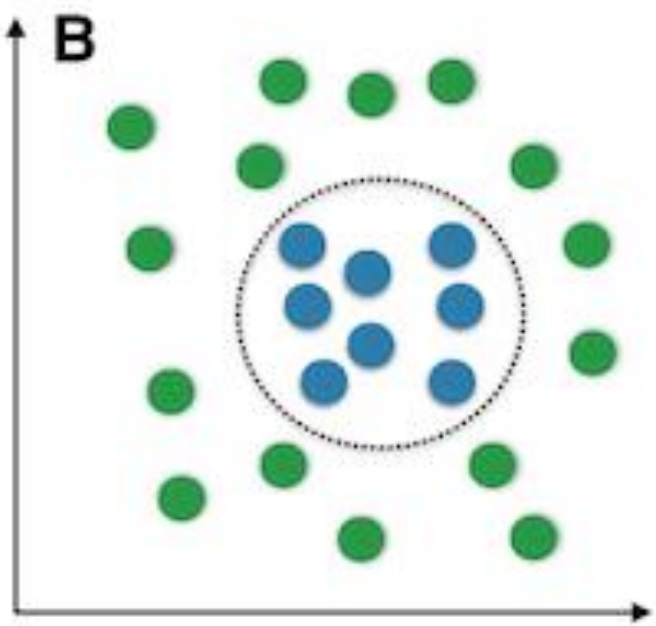
Linear SVM

- Data is linearly separable
- Difficult for more than two Classes



Non-Linear SVM

- Data is not linearly separable
- Best for two or more than two Classes

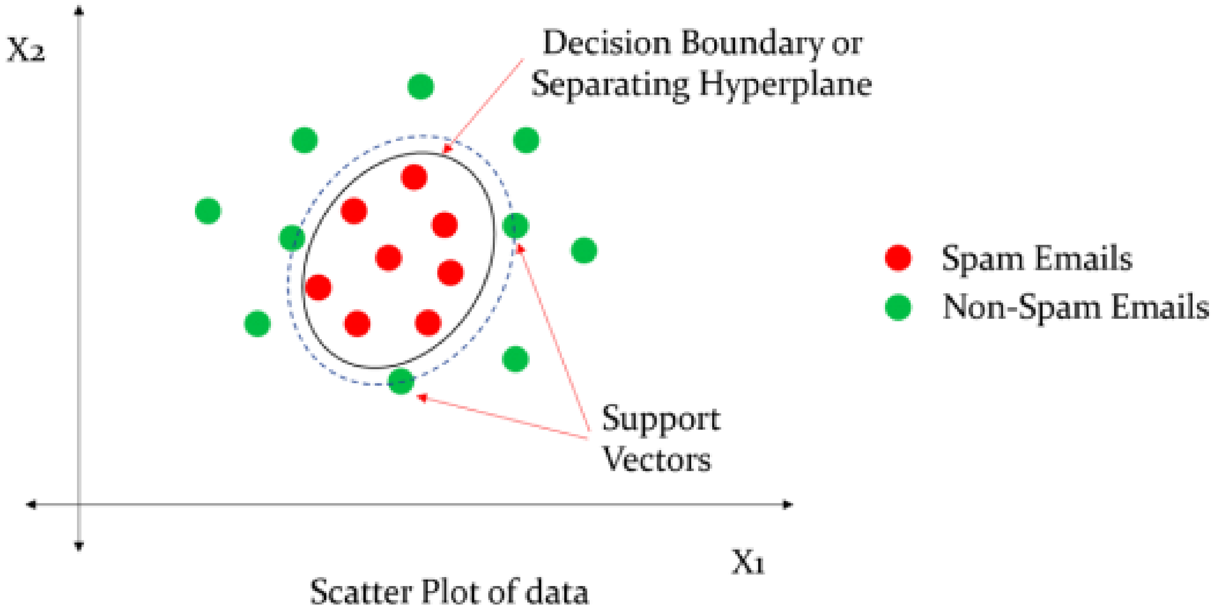


Source: https://tinyurl.com/yyekewju

Non-Linear SVMs

- Cannot be separated by a linear line
- Transforms into higher dimensions & making it linearly separable

Non-linear decision boundary in the case of the Email Spam classification example.



Source: https://tinyurl.com/y8wy6p27

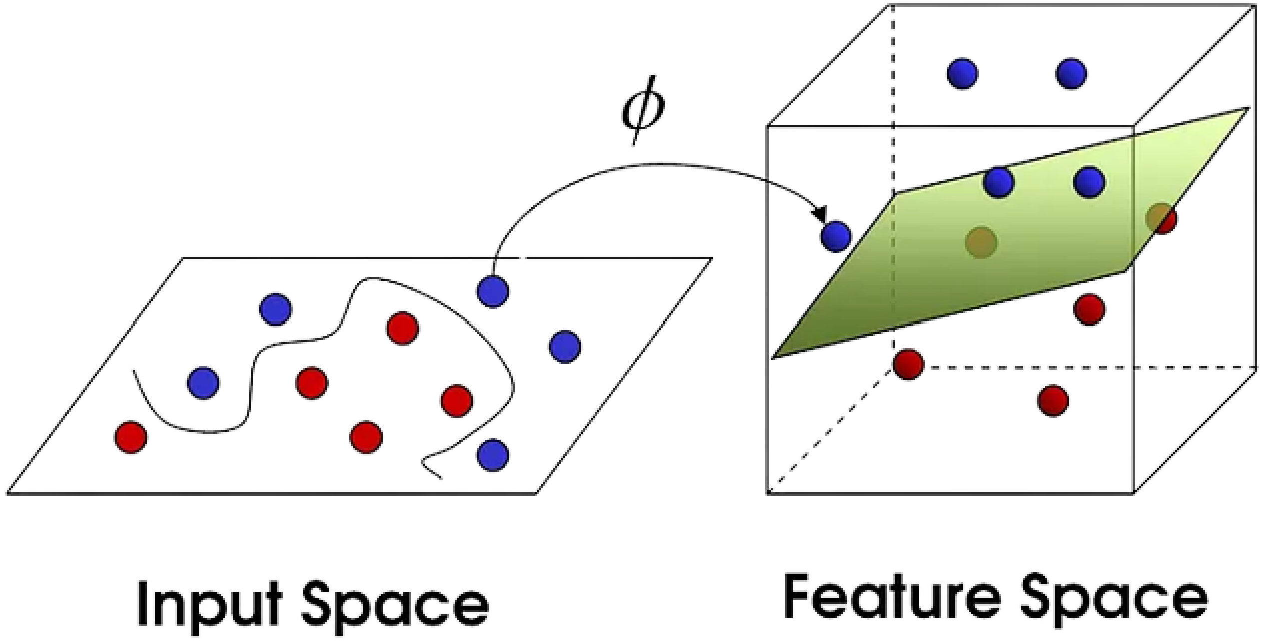
Important Terminologies

- **Hyperplane**: Decision boundary - to separate the data points of different classes.
- **Support Vectors**: Data points closest to the decision boundary
- **Margin**: Distance between the decision boundary and the support vectors
- **Kernel**: Mathematical function - transform into high-dimensional feature spaces. (Linear, polynomial, Radial basis function (RBF) and sigmoid)
- **Nonlinear Decision Boundaries**: Complex decision boundaries - separate data points accurately
- **Regularization Parameter (C)**: Controls the trade-off between the misclassification of training examples and the margin width.

Non-Linear SVM: Working

- Non-Linear to Linear space
  (2D to 3D transformation)
- Finds the right Hyperplane
  (Which has the maximum margin)

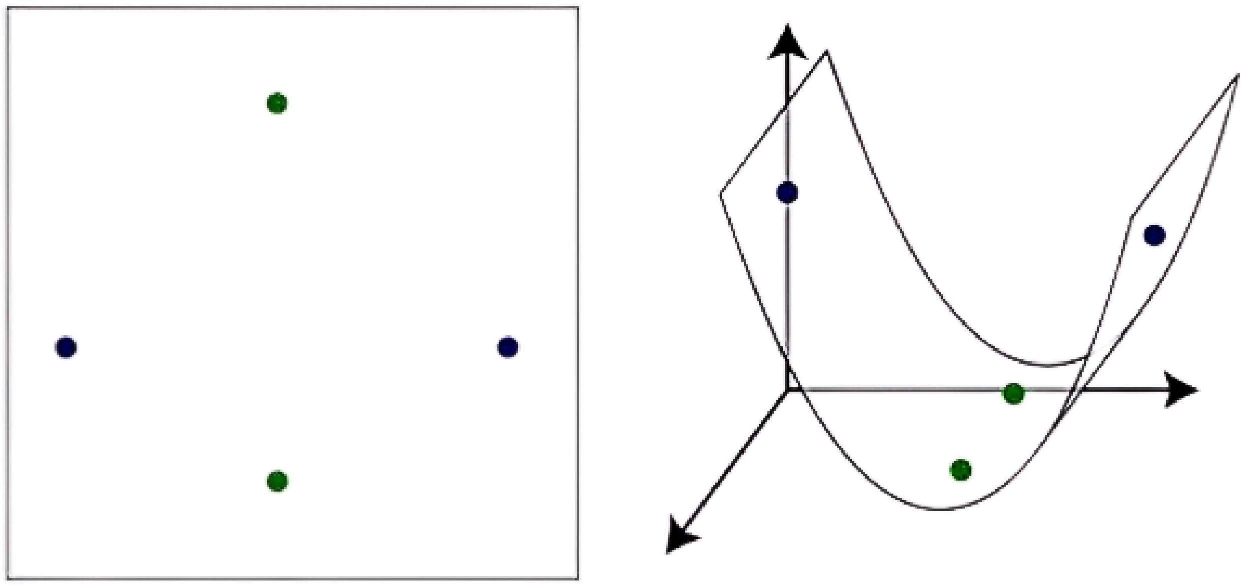
Transformation of data points to higher dimensions



Source: https://tinyurl.com/bdcrdssz

Kernel Trick

Method to project non-linear data onto higher dimensions.



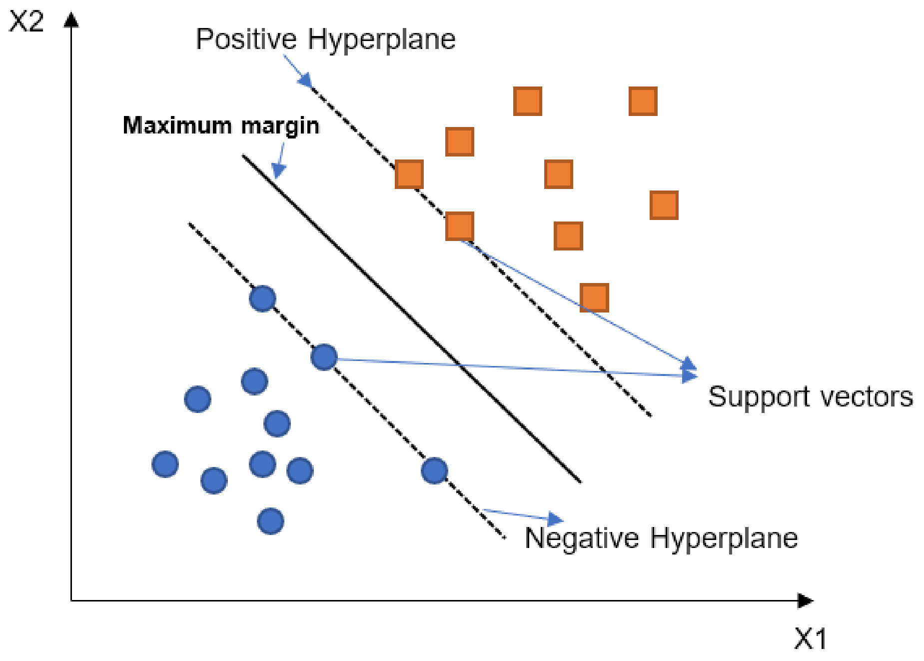
- Kernel Functions –
  - Linear
  - Polynomial
  - Gaussian radial basis function (RBF)
  - Sigmoid

Source: https://tinyurl.com/33khdvps

Linear Kernel

- Linear Separability: separated by a single straight line.
- Efficient Processing: computationally efficient, especially with 1-D features

Linear kernel for linearly separable data points

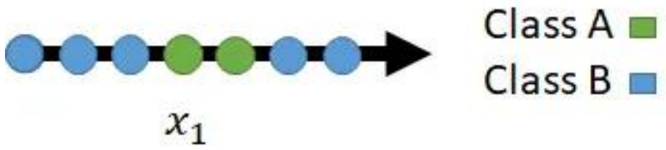


Source: https://tinyurl.com/3r8cbjdk

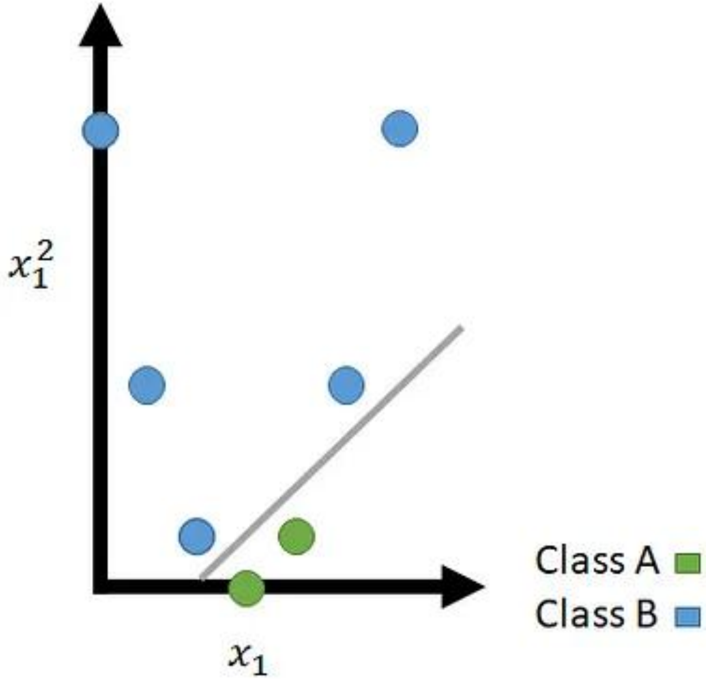
Polynomial Kernel

Polynomial Kernel – when data can be separated by some polynomial function.

Non-linear to linear transformation using polynomial kernel functions



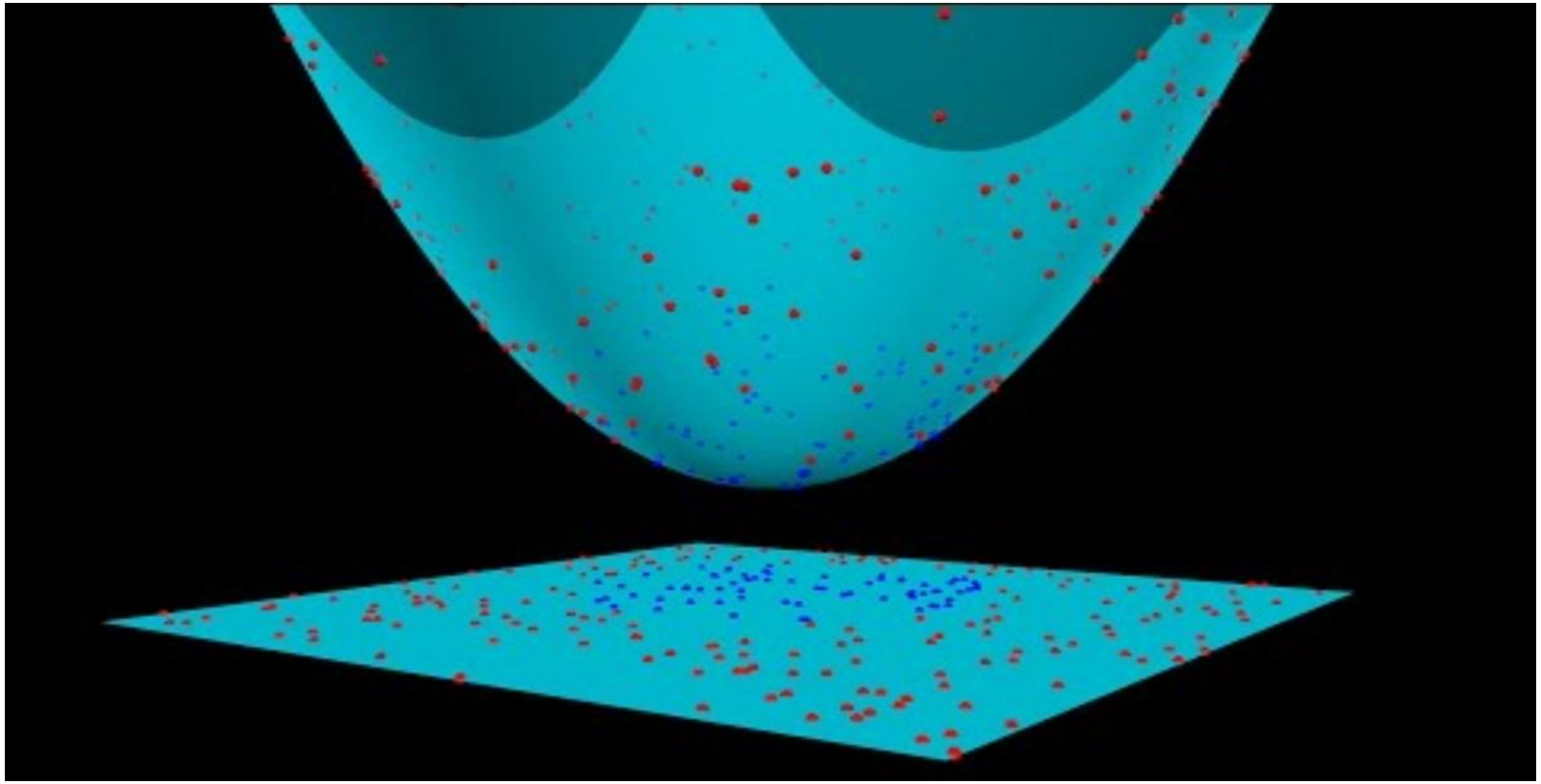
x_1	Class
-3	B
-2	B
-1	B
0	A
1	A
2	B
3	B



x_1	x_1^2	Class
-3	9	B
-2	4	B
-1	1	B
0	0	A
1	1	A
2	4	B
3	9	B

Source: https://tinyurl.com/bdfr2hte

Polynomial Kernel Visualization

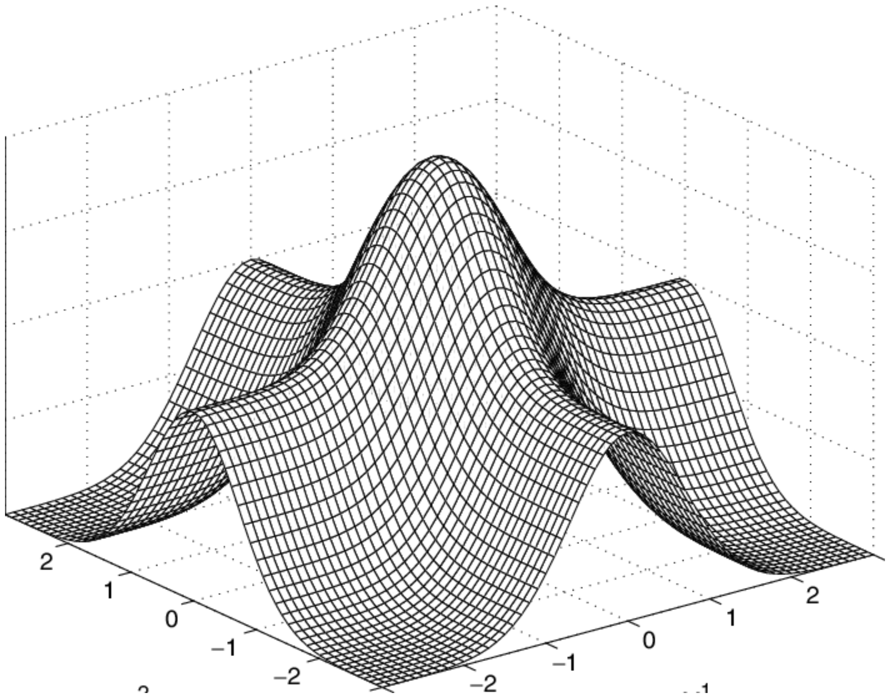


Source: https://youtu.be/OdlNM96sHio?si=CKDR-N_EntkYS9Q0

RBF Kernel

- More complex & efficient
- Can combine multiple polynomial kernels multiple time of different degrees.

Transformed version using RBF kernel



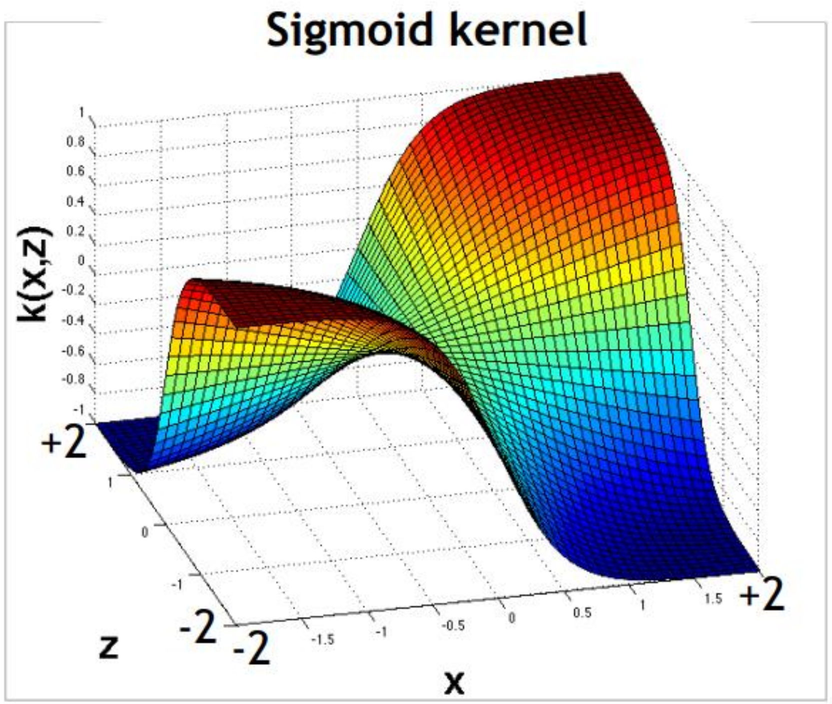
Source: https://doi.org/10.1007/10984697_3

University of Windsor

Sigmoid Kernel

- Can capture complex non-linear data patterns.
- Versatile and suitable for addressing data with unknown characteristics

Transformed version using Sigmoid kernel



Source: https://tinyurl.com/3x4cmamu

Mathematics: Linear Kernel Function

1. Linear Kernel function

- Linear Kernel is an SVM kernel function that is used for data that is linearly separable, making it an effective choice for simple classification problems. It calculates the dot product between feature vectors in their original space.
- The formula for the linear kernel is as follows:

$$\mathrm{f}(X, Y) = X^T . Y$$

- $\mathrm{f}(X, Y)$ represents the kernel function, which takes two input vectors, X and Y.
- X^T is the transpose of X vector and Y is the second vector.
- The dot product (.) between X^T and Y is used to compute the similarity between the two input vectors.

Mathematics: Polynomial Kernel Function

2. Polynomial Kernel function

- The formula for the polynomial kernel is as follows:

$$f(X_1, X_2) = (X_1^T . X_2 + 1)^d$$

  - Here, X_1 and X_2 are input feature vectors that indicate data points.
  - $X_1^T \cdot X_2$ is the dot product of two feature vectors, where X1T is the transpose of the first vector and X2 is the second vector.
  - d is the degree of the polynomial.

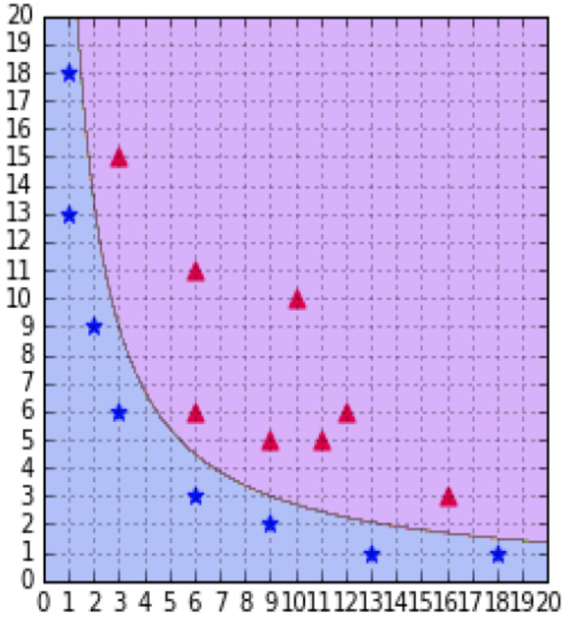
- Assuming we have the two features X_1 and X_2 and Y as the output variable, we can define it as follows using the polynomial kernel:

$$\mathrm{X}_1^{\mathrm{T}} . \mathrm{X}_2 = \begin{bmatrix} \mathrm{X}_1 \\ \mathrm{X}_2 \end{bmatrix} . [\mathrm{X}_1 \ \mathrm{X}_2] = \begin{bmatrix} \mathrm{X}_1^2 & \mathrm{X}_1 . \mathrm{X}_2 \\ \mathrm{X}_1 . \mathrm{X}_2 & \mathrm{X}_2^2 \end{bmatrix}$$

Mathematics: Polynomial Kernel Function

- As a result, 2 dimensions were transformed into 5 dimensions.

SVM using a polynomial kernel is able to separate the data (degree=2)



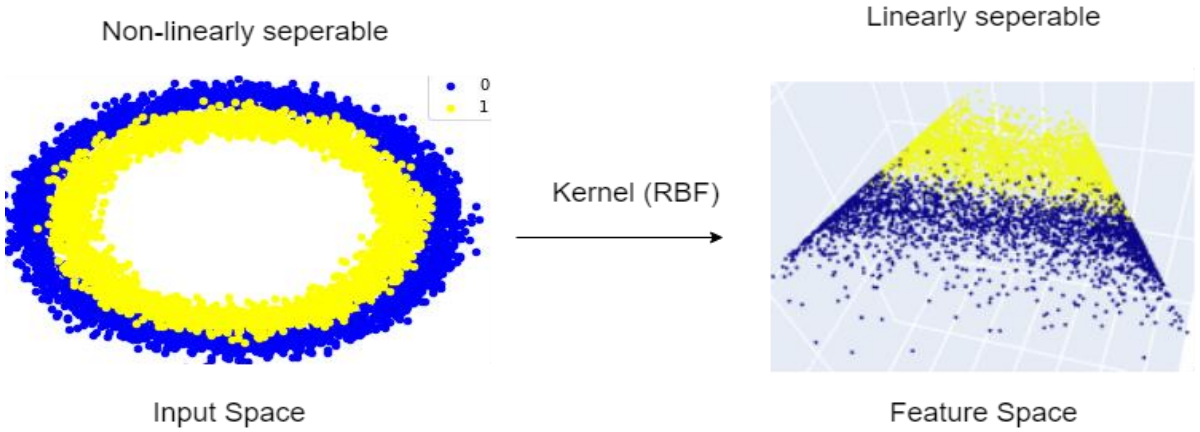
Source: https://tinyurl.com/msxashrw

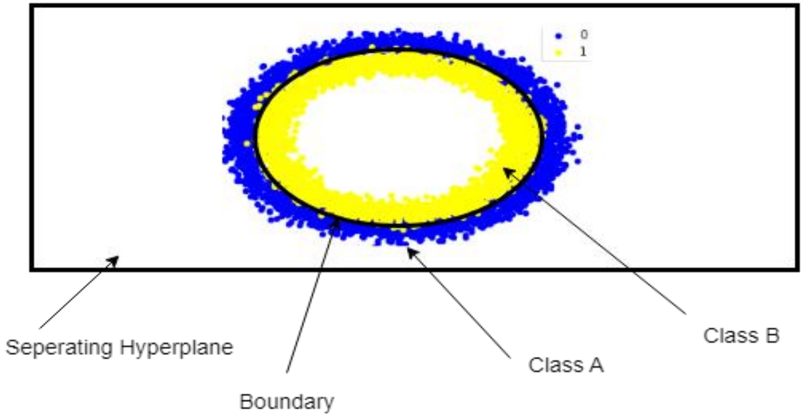
Mathematics: RBF Kernel

3. Radial Basis Function Kernel

- The RBF kernel works by translating the data into a high-dimensional space and then doing classification using the core principle of Linear SVM.
- The radial basis function is represented by the following formula:

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





Source: https://tinyurl.com/y4fd4vc

University of Windsor

Mathematics: RBF Kernel

- Here, $||X_1 - X_2||^2$: This part calculates the squared Euclidean distance between the two input vectors, x and y.
- σ is a free parameter that could be used to tune the equation.

- If a new parameter $\gamma = 1 / 2\sigma^2$ is added, then the equation will be

$$K(X_1, X_2) = exp(-\gamma||X_1 - X_2||^2)$$

- This equation helps us measure how similar different data points are without having to do a lot of complicated calculations. It's a quick and easy way to compare data points and see how closely related they are.

Mathematics: Sigmoid Kernel Function

4. Sigmoid Kernel function

- The formula for the Sigmoid kernel is as follows:

$$f(X, Y) = (\alpha . X^T . Y + C)$$

  - $f(X, Y)$ denotes the kernel function value for two input data points X and Y.
  - α is a user-defined parameter that controls the steepness of the hyperbolic tangent (tanh) function. It must be a positive constant.
  - X^T is the transpose of X vector and Y is the second vector.
  - C is a constant that can also be chosen by the user.

- The sigmoid kernel is different from other kernel functions. It employs the hyperbolic tangent function to translate data into a higher-dimensional space, producing an "S"-shaped decision boundary that can capture complex non-linear connections.
- Also, S"-shaped curve that maps its input values into the range of [-1, 1].

Evaluation Metrics

- Hinge Loss (Loss Function):
  - Measures classification performance in SVM.
  - Encourages a wide margin between different classes.
- Hinge Loss = max(0, 1 - y*f(x))
  - y is the true label.
  - f(x) is the model's prediction.
- Penalty Term (Regularization):
  - Control overfitting.
  - Adjusted by hyperparameter C.
- Cost = Hinge Loss + C * Regularization Term
  - C = Controls balance between margin and classification errors

- In nonlinear SVMs, a kernel function (e.g., RBF, polynomial) indicates data to a higher-dimensional space for linear separation.
- The hinge loss and penalty term operate in this transformed area.

Numerical Example

Numerical example for Non-Linear SVM:

- Let's suppose we are provided with the positively labelled data points below:

$$\left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}2\\-2\end{pmatrix},\begin{pmatrix}-2\\-2\end{pmatrix},\begin{pmatrix}-2\\2\end{pmatrix}\right\}$$

$$\left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}2\\-2\end{pmatrix},\begin{pmatrix}-2\\-2\end{pmatrix},\begin{pmatrix}-2\\2\end{pmatrix}\right\} \longrightarrow \left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}10\\6\end{pmatrix},\begin{pmatrix}6\\6\end{pmatrix},\begin{pmatrix}6\\10\end{pmatrix}\right\}$$

- And negatively labelled data points below:

$$\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$$

$$\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\} \longrightarrow \left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$$

Numerical Example

- We need to identify a hyper-plane, or a line that separates data into two groups. However, because this is a non-linear SVM, Kernal SVM must be used to transfer data from one feature space to another.
- Kernel SVM condition:

$$\phi_1\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{cases}\begin{pmatrix}4-x_2+|x_1-x_2|\\4-x_1+|x_1-x_2|\end{pmatrix} & if\sqrt{X_1^2+X_2^2} > 2\\ \begin{pmatrix}x_1\\x_2\end{pmatrix} & \end{cases}$$

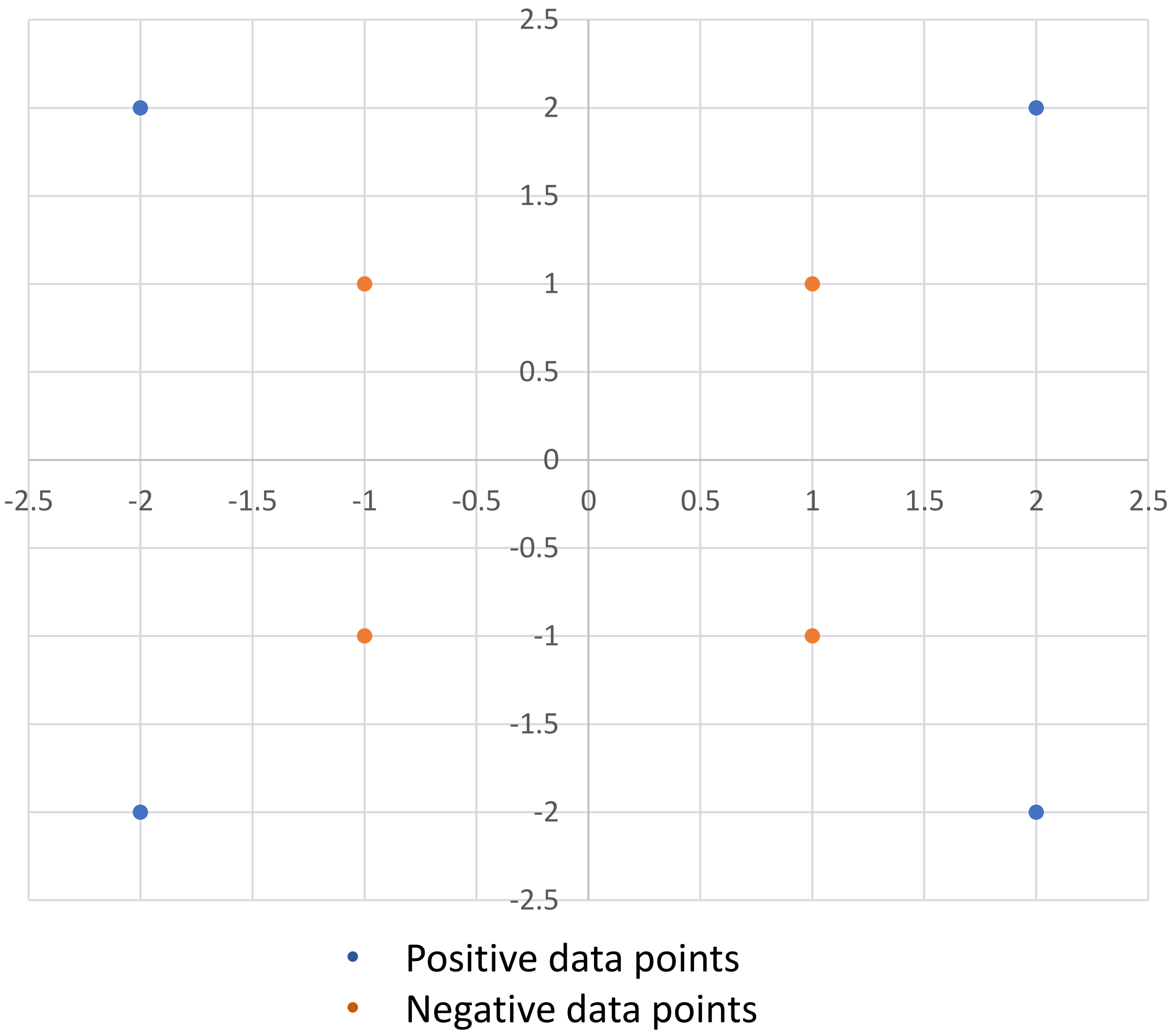
- Both positive and negative data points are applied to the condition.

Positive data points:

$$\left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}2\\-2\end{pmatrix},\begin{pmatrix}-2\\-2\end{pmatrix},\begin{pmatrix}-2\\2\end{pmatrix}\right\} \longrightarrow \left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}10\\6\end{pmatrix},\begin{pmatrix}6\\6\end{pmatrix},\begin{pmatrix}6\\10\end{pmatrix}\right\}$$

Negative data points:

$$\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\} \longrightarrow \left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$$



Numerical Example

- Based on these data points, if we put it on the plot figure, we can easily find a hyperplane which can divide data points into 2 parts.
- Now, we can easily identify the support vectors.

$$\left\{s_1 = \begin{pmatrix}1\\1\end{pmatrix}, s_2 = \begin{pmatrix}2\\2\end{pmatrix}\right\}$$

- Each vector is augmented with a 1 as a bias input.

$$\tilde{s_1} = \begin{pmatrix}1\\1\\1\end{pmatrix} \qquad \tilde{s_2} = \begin{pmatrix}2\\2\\1\end{pmatrix}$$

$$\alpha_1\ \tilde{s_1}.\ \tilde{s_1} + \alpha_2\ \tilde{s_2}.\ \tilde{s_1} = -1$$

$$\alpha_1\ \tilde{s_1}.\ \tilde{s_2} + \alpha_2\ \tilde{s_2}.\ \tilde{s_2} = +1$$

$$\alpha_1 \begin{pmatrix}1\\1\\1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\2\\1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix} = -1$$

$$\alpha_1 \begin{pmatrix}1\\1\\1\end{pmatrix}\begin{pmatrix}2\\2\\1\end{pmatrix} + \alpha_2 \begin{pmatrix}2\\2\\1\end{pmatrix}\begin{pmatrix}2\\2\\1\end{pmatrix} = 1$$

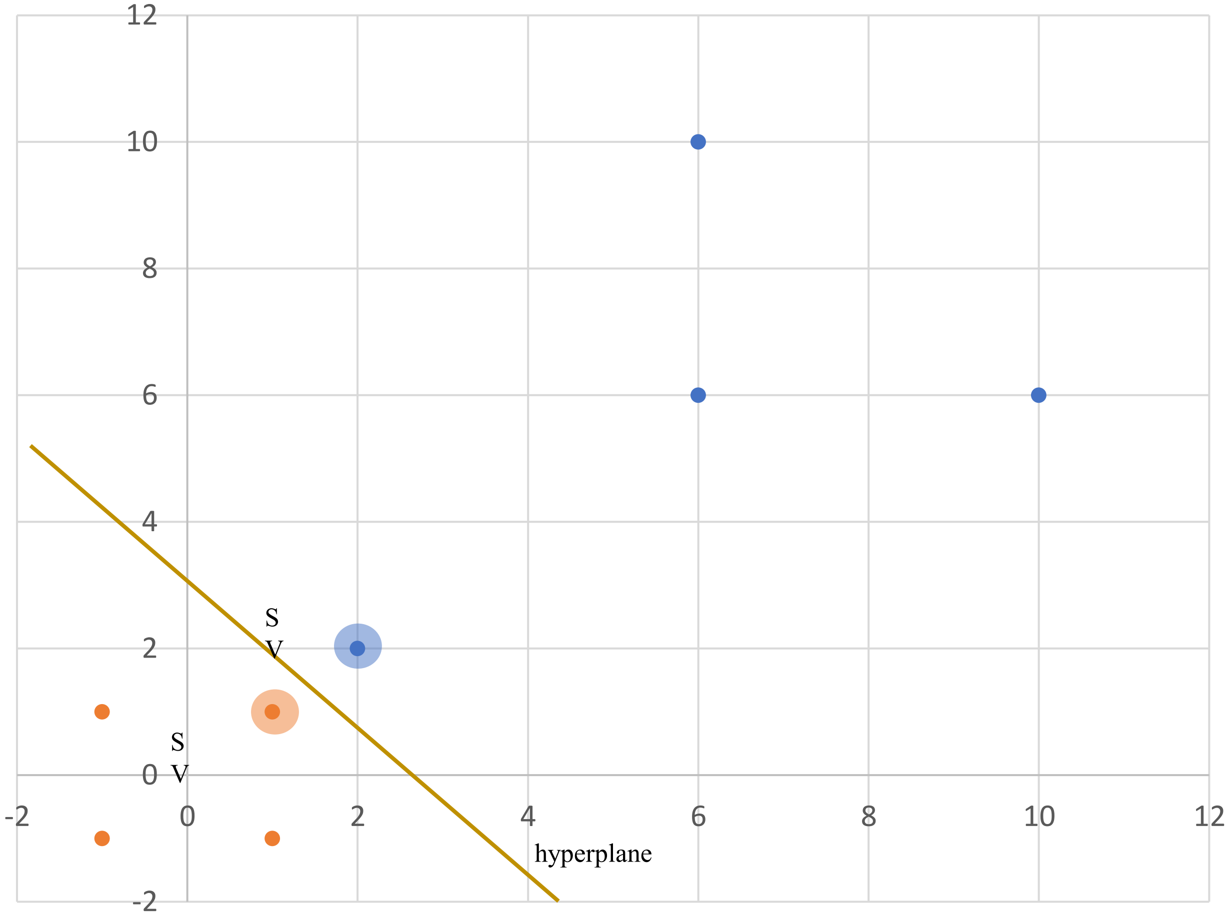
$$\alpha_1\ (1 + 1 + 1) + \alpha_2\ (2 + 2 + 1) = -1$$

$$\alpha_1\ (2 + 2 + 1) + \alpha_2\ (4 + 4 + 1) = 1$$

$$3\alpha_1 + 5\alpha_2 = -1$$

$$5\alpha_1 + 9\alpha_2 = 1$$

$$\alpha_1 = -7 \qquad \alpha_2 = 4$$



Source: https://tinyurl.com/eaarrnxr

Numerical Example

$$\widetilde{\omega} = \sum_i \alpha_i \, \widetilde{s_1} = -7\begin{pmatrix}1\\1\\1\end{pmatrix} + 4\begin{pmatrix}2\\2\\1\end{pmatrix}$$

$$= \begin{pmatrix}1\\1\\-3\end{pmatrix}$$

- Lastly, keep in mind that a bias has been added to our vectors.
- The final input in $\widetilde{\omega}$ can also be written as the separating operator and the hyperplane offset b.
- Hyperplane equation y $= \widetilde{\omega}$x+b
- with $\widetilde{\omega} = \begin{pmatrix}1\\1\end{pmatrix}$ and b = -3

University of Windsor

Evaluation Metrics

❖ Mean Absolute Error (MAE):

- It determines the mean absolute difference between the actual and anticipated values. In nonlinear SVM, it indicates the average absolute error in the model's predictions.

Mean Absolute Error= (1/n) * $\sum_{i=1}^{n} |A_i - P_i|$

- Here, A_i = Actual value, P_i = Prediction value.
- MAE is the average sum of all absolute errors [10].
- Prediction error=Actual value A_i -Predicted value P_i
- Absolute error=|Prediction error(P_i)|

❖ Mean Squared Error (MSE):

$$\text{MSE} = \frac{\sum_{i=1}^{n}(A_i - P_i)^2}{n}$$

- MSE measures the average squared difference between the actual and predicted values.
- It measures the average squared error of the SVM's predictions in the context of the SVM.

Evaluation Metrics

❖ Root Mean Squared Error (RMSE):

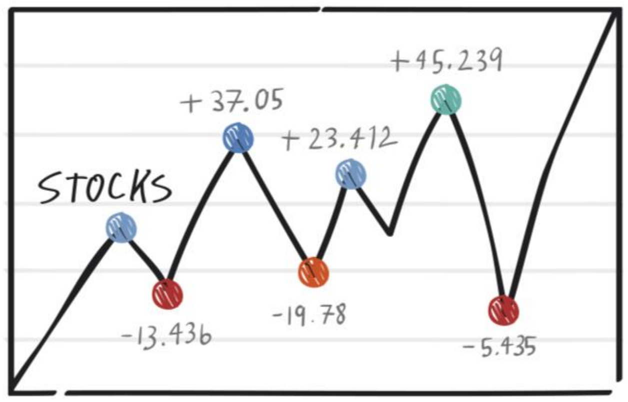
- RMSE is the square root of the MSE.
- It measures the average squared error of the SVM's predictions in the context of the SVM.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(A_i - P_i)^2}{n}}$$

❖ R-squared (R^2):

- R-squared, commonly known as the coefficient of determination.
- The range of the R-squared value is 0 to 1. If the model does not match the algorithm, the result will be negative. If the value is greater than or equal to one, it is the better model.

SVM: Applications

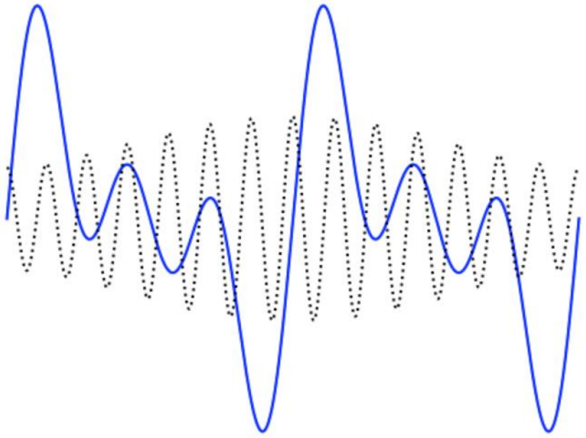


Financial Forecasting & Fraud Detection

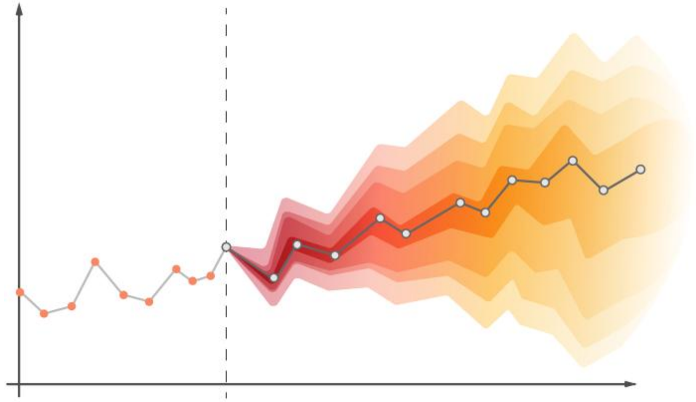
Natural Language Processing

Image Recognition & Classification

Medical Diagnosis



Signal Processing



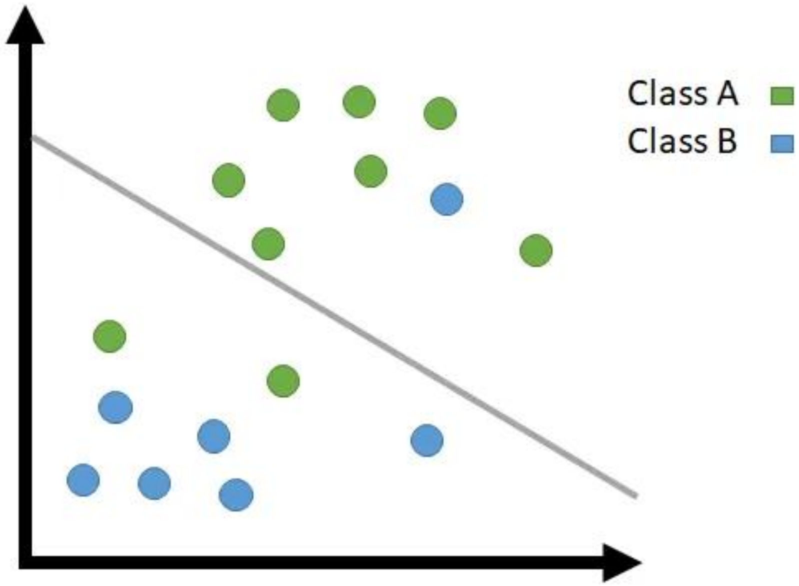
Time Series Forecasting

Implementation: Intuitive Example

➢Goal: Determine whether the fruit is Ripe or Not.

- Linear Case: Based on Color
- Non-Linear Case: Color, firmness, and scent

Linear Classification Plot



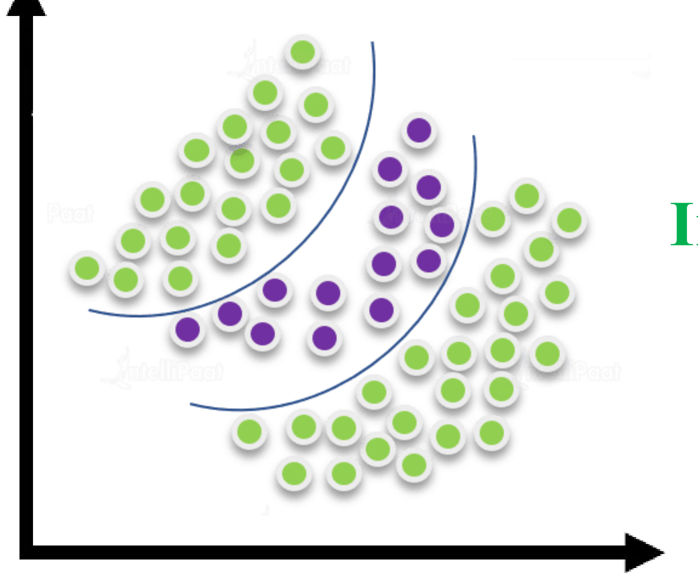
Source: https://tinyurl.com/bdfr2hte

University of Windsor

Implementation: Intuitive Example

- ➢Goal: Determine whether the fruit is Ripe or Not.
- Non-Linear SVM decides boundary that considers:
  - ➢Interactions & combinations of features
  - ➢Identifying complex patterns
  - ➢Adapting to variations in ripeness.

Non-Linear Classification Plot



Inherently a non-linear problem!

Source: https://tinyurl.com/yvhy5eyz

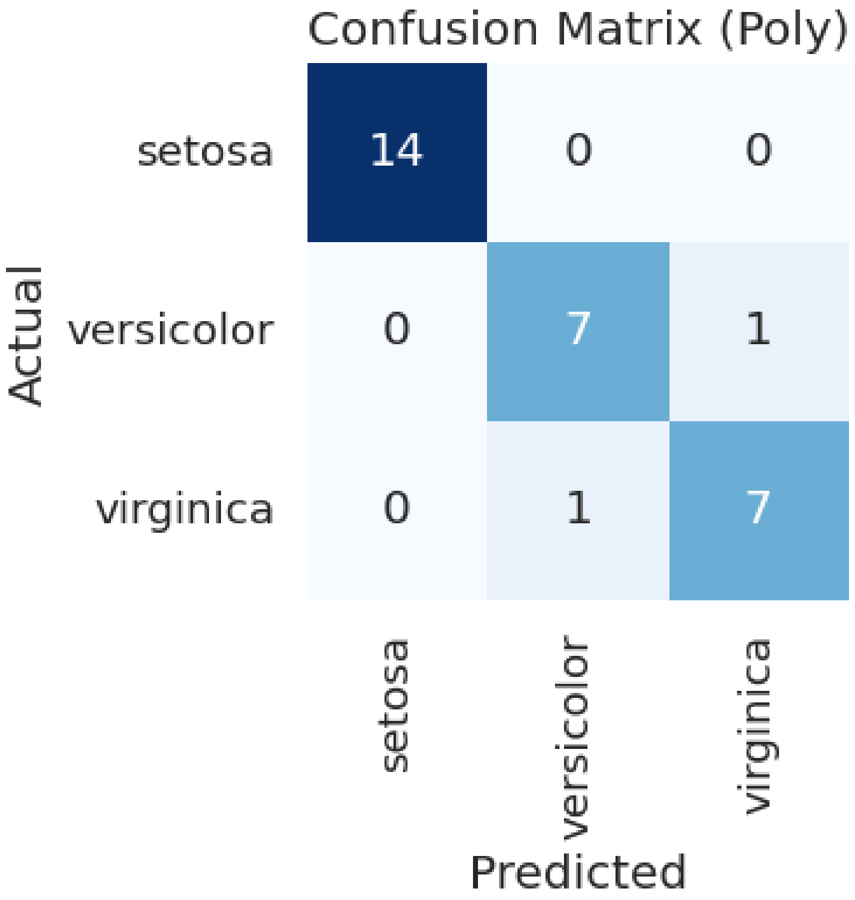
Implementation: Code

```
# <== Importing Necessary Libraries ==>
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,
classification_report, accuracy_score
import seaborn as sns

# <== Load the IRIS Dataset ==>
iris = load_iris(as_frame=True)
df = iris.frame
X = df[['sepal length (cm)','petal width (cm)']].values
y = df.target
```

```
# <== Split the dataset into training and testing sets ==>
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=2)
# <== Create a non-linear kernel SVM classifier ==>
svm = SVC(kernel='poly', degree=5, random_state=42)
svm.fit(X_train, y_train)

# <== Make predictions on the test data ==>
y_pred = svm.predict(X_test)

# <== Calculate confusion matrix ==>
confusion = confusion_matrix(y_test, y_pred)
```

Implementation: Code

The resulting confusion matrix is as:



Class-wise model evaluation metrics:

Classes	Precision	Recall	F1-Support
setosa	1.00	1.00	1.00
versicolor	0.88	0.88	0.88
virginica	0.88	0.88	0.88

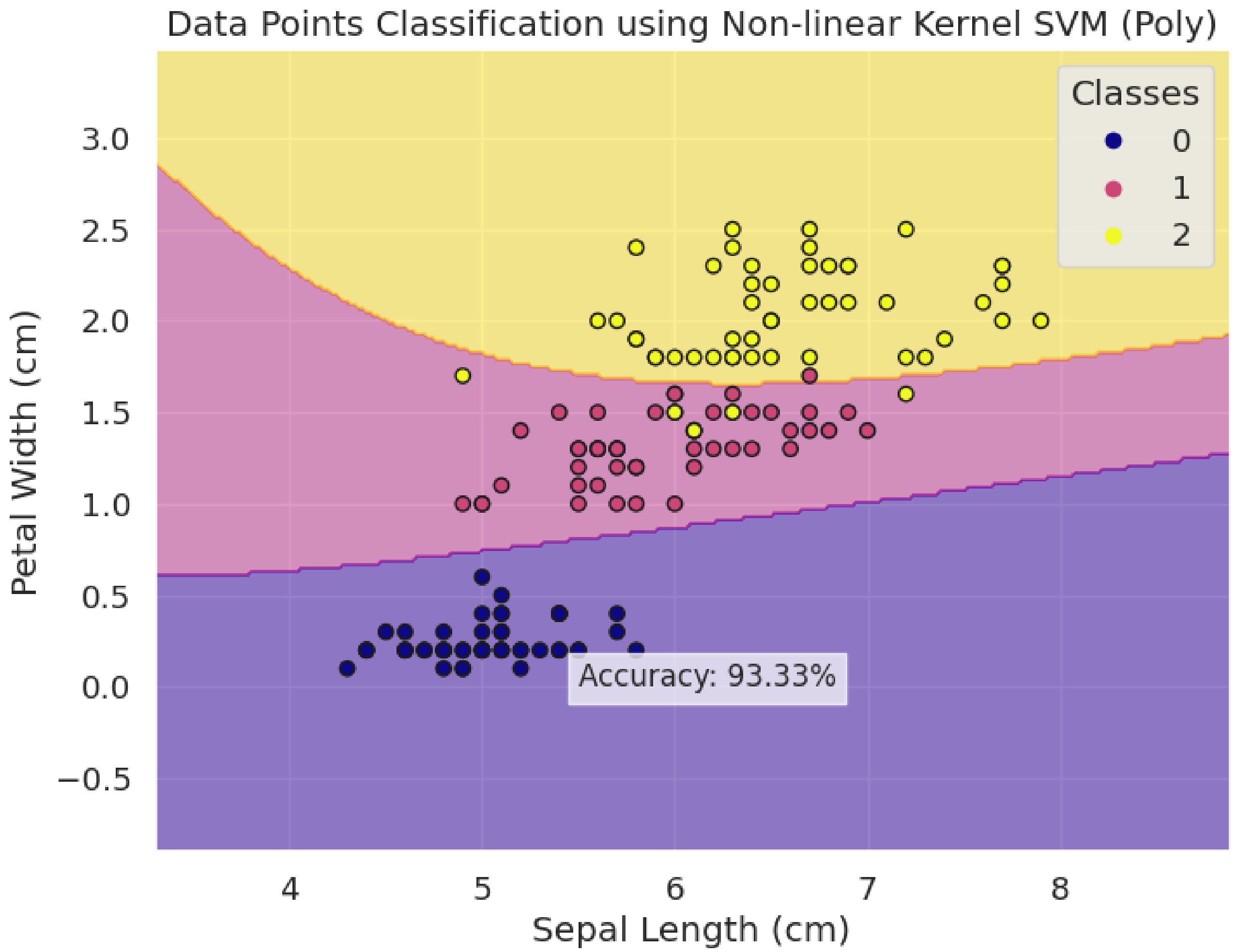
Implementation: Decision Boundary

```
# <== Plotting the decision boundary ==>
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
Z = svm.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)


plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.5, cmap='plasma')
scatter = plt.scatter(X[:, 0],
    X[:, 1], c=y, edgecolors='k', cmap='plasma')
```

```
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('Data Points Classification using Non-linear
    Kernel SVM (Poly)')
legend
    = plt.legend(*scatter.legend_elements(), title="Classes")


# <== Calculating the accuracy ==>
y_pred = svm.predict(X_test)
accuracy = (y_pred == y_test).mean()
plt.text(5.5, 0, f'Accuracy: {accuracy
    * 100:.2f}%', fontsize=12, bbox=dict(facecolor='white', a
    lpha=0.7))
plt.show()
```

Implementation: Decision Boundary

The resulting decision boundary with accuracy is as:



University of Windsor

Implementation: Code

Code Link: http://tiny.cc/non-linear-svm-example

Non-Linear SVM: Comparision

Advantages

1. Handling Non-Linearity
2. High Accuracy
3. Robust to Outliers
4. Effective in High Dimensional
5. Wide Range of Applications

Limitations

1. Computationally Intensive
2. Parameter Sensitivity
3. Limited Scalability
4. Prone to Overfitting
5. Choosing Kernel Functions

Overcoming Limitations

1. **Computationally Intensive:** Dimensionality Reduction, Efficient Libraries
2. **Parameter Sensitivity:** Grid Search, Regularization
3. **Limited Scalability:** Linear Approximations, Parallelization
4. **Prone to Overfitting:** Feature Selection, Regularization
5. **Choosing Kernel Functions:** Cross-Validation, Kernel Parameters

Conclusion

- **Overview:** Linear vs Non-Linear SVMs
- **Kernel Functions:** Types, Underlying Concepts
- **Mathematical Equations:** Algorithm and Hyperparameters
- **Evaluation Metrics:** Assess model behavior
- **Implementation & Applications**
- **Benefits**
  - ➢ **Versatile & Powerful:** Complex classification problems.
  - ➢ **Excels in Non-linear relationships:** Suitable for a wide range of applications.
  - ➢ **Choice of Kernels & Hyperparameters:** Crucial for Optimal Performance

Other Important Links

https://www.youtube.com/watch?v=DB6fCUBKiKs

https://www.youtube.com/watch?v=qjErFs0tepw

https://www.youtube.com/watch?v=bwrKtkW2RdI

https://www.youtube.com/watch?v=03IrkMM4E6M

References

[1] S. Arora, “SVM: Difference between Linear and Non-Linear Models,” AITUDE, Feb. 04, 2020. https://www.aitude.com/svm-difference-between-linear-and-non-linear-models/

[2] M. Ahmed, “Non-linear Support Vector Machines Explained,” Medium, May 21, 2022. https://linguisticmaz.medium.com/support-vector-machines-explained-ii-f2688fbf02ae

[3] B. Gupta, “Non-Linear SVM,” Scaler Topics, Feb. 02, 2023. https://www.scaler.com/topics/machine-learning/non-linear-svm/

[4] “Nonlinear Support Vector Machine - an overview | ScienceDirect Topics,” www.sciencedirect.com. https://www.sciencedirect.com/topics/engineering/nonlinear-support-vector-machine

[5] V. Piccialli and M. Sciandrone, “Nonlinear optimization and support vector machines,” Annals of Operations Research, vol. 314, no. 1, pp. 15–47, Apr. 2022, doi: https://doi.org/10.1007/s10479-022-04655-x.

[6] “Linear SVM vs Nonlinear SVM high dimensional data,” Stack Overflow. https://stackoverflow.com/questions/44606126/linear-svm-vs-nonlinear-svm-high-dimensional-data

Thank You!