Support Vector Machines (SVM)

About Me

- Shubhan Tamhane
- Sophmore studying Statistical Data Science and minor in Financial Analysis
- **Motivation:** SVM is a very popular ML Algorithm and I wanted to learn how it works in terms of classification

---

Introduction

- **What is SVM?**
  A supervised machine learning algorithm used mainly for **classification**
- **Why use it?**
  Works well in high-dimensional spaces and finds a clear decision boundary between classes

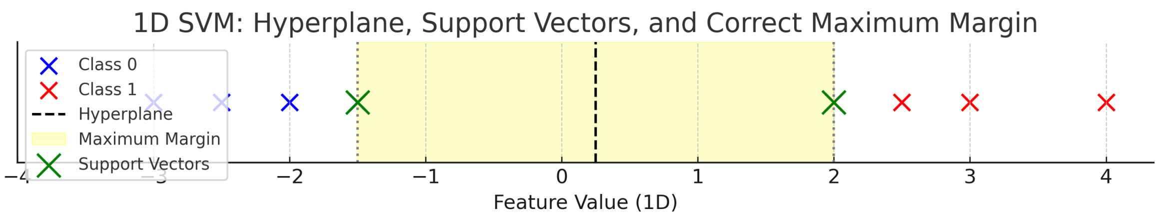
---

Core Idea

- SVM finds the **best boundary** (line or hyperplane) that separates data into classes
- It chooses the one with the **maximum margin** — the widest gap between the classes
- The closest data points to the boundary are called **support vectors**

What This Means

- **Example:** Suppose we have a dataset of Class 1 and Class 0
- Very separable
- We can see the hyperplane, maximum margin, and the support vectors



–

Key Concepts

Margin

- Distance between decision boundary and nearest data points

- SVM **maximizes** this gap

Support Vectors

- Closest points to the margin

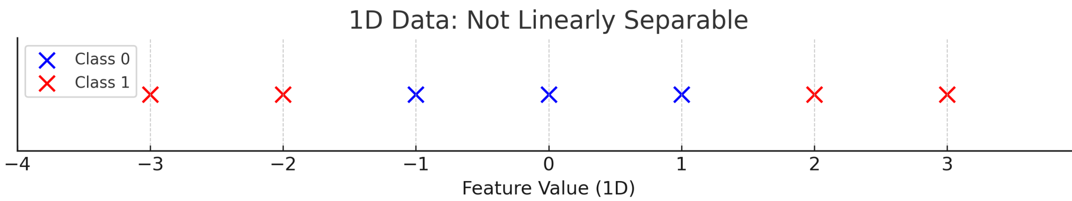
- Define the decision boundary

- Only these matter for SVM's learning

–

More Realistic Example

- In this example, we can see that it is impossible to set a hyperplane without having many misclassifications due to high overlap



- So what can we do?

–

More Realistic Example contd.

- To fix this we can plot this in a two-dimensional space
  with the X-value as the original data point and
  the Y-value as the square of the original point
- To code this in Python we would say:

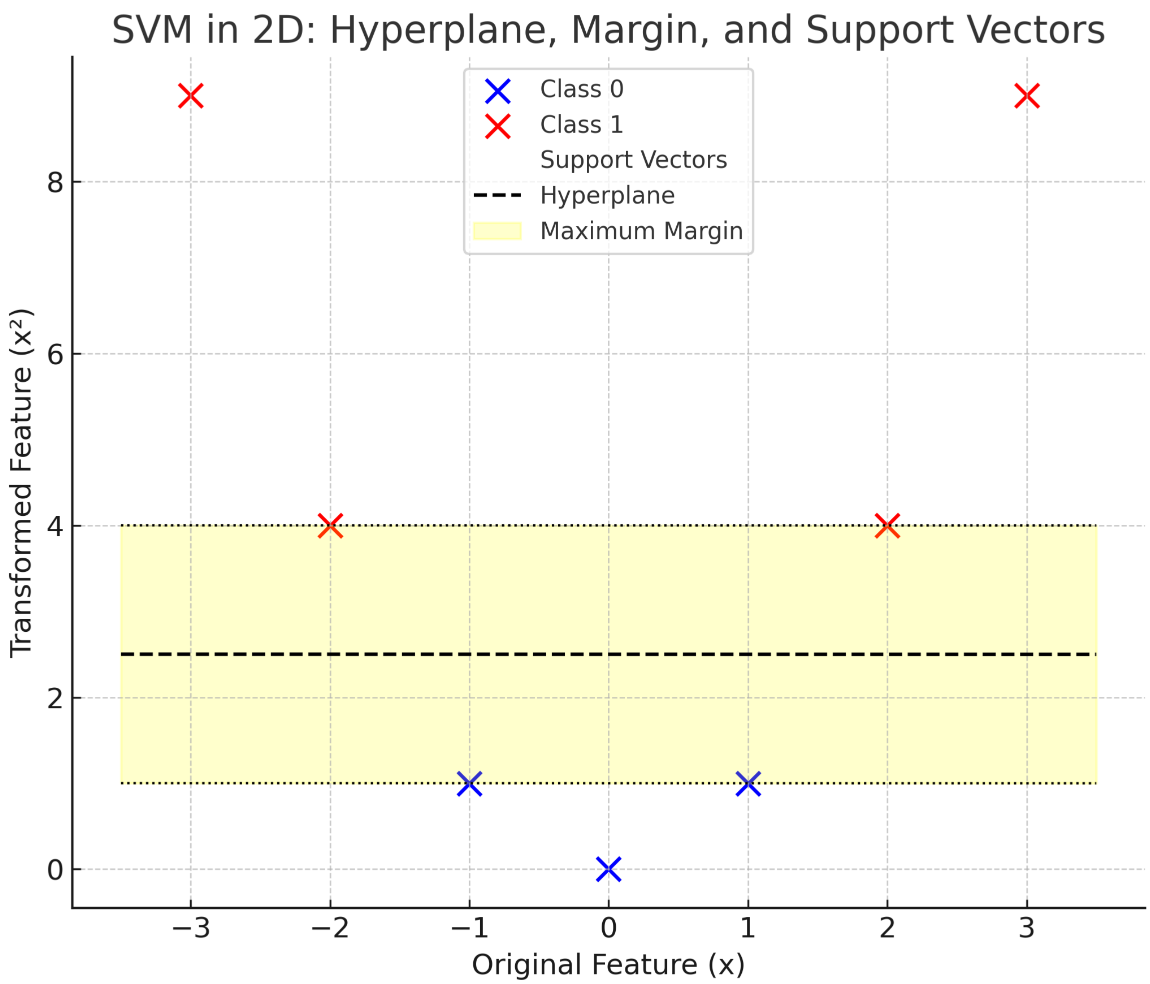
```
from sklearn.svm import SVC

model = SVC(kernel = 'poly', degree = 2)
```

- Later, we'll see how SVM can do this transformation automatically using something called a **kernel**

–

More Realistic Example contd.



- Transformed data is now **linearly separable**
- SVM finds a **linear boundary** in 2D

–

Linear vs Non-Linear

- **Linearly separable**: A straight line (or plane) can separate the classes (Example 1)
- **Non-linear data**: Use **kernels** to transform the space (Example 2)

The Kernel Trick

- Kernels transform the data into higher-dimensional space **without computing the transformation directly**
- When we mapped out x^2, we did it manually, but the Kernel trick can do this for us automatically
- This allows SVM to draw a straight boundary in the transformed space

```
Real-world analogy:
Imagine red and blue sprinkles arranged in a ring pattern. In 2D you can't separate them, bu
```

Common Kernel Types

Kernel	Use Case
Linear	Data that's already separable with a straight line (Example 1)
Polynomial	When the decision boundary is curved (Example 2)
RBF (Radial Basis Function)	For complex, non-linear boundaries

RBF measures similarity based on distance — closer points are more related.

Radial Basis Function Kernel

- RBF kernel flexibly bends the boundary to fit the shape
- Measures similarity between two points based on distance
- Maps data points in an infinite-dimensional feature space
- Flexible and powerful

Parameters

Parameter	Description	Example
`kernel`	Defines the shape of the decision boundary	`'linear'`, `'rbf'`, `'poly'`
`C`	Controls trade-off between margin size and misclassification	`C = 0.1` (wide margin), `C = 100` (strict)
`gamma`	Defines how far the influence of a point reaches	`'scale'`, `0.1`, `1`
`degree`	Degree of the polynomial kernel	`degree = 2` or `3`

Pros & Cons

Pros	Cons
Works well in high-dimensional space	Computationally expensive on large datasets
Effective with small datasets	Sensitive to parameter tuning
Prevents overfitting well	Not great with overlapping groups

Real-World Applications

- Face recognition
  - **Facebook and Apple**
- Spam email detection
- Speech Recognition
  - **Siri and Google**
- Image Classification
  - **Pintrest**

Summary

- SVM finds a decision boundary with the **maximum margin**
- **Support vectors** define the boundary
- **Kernels** allow SVM to work with complex data shapes

Thank You!

Questions?