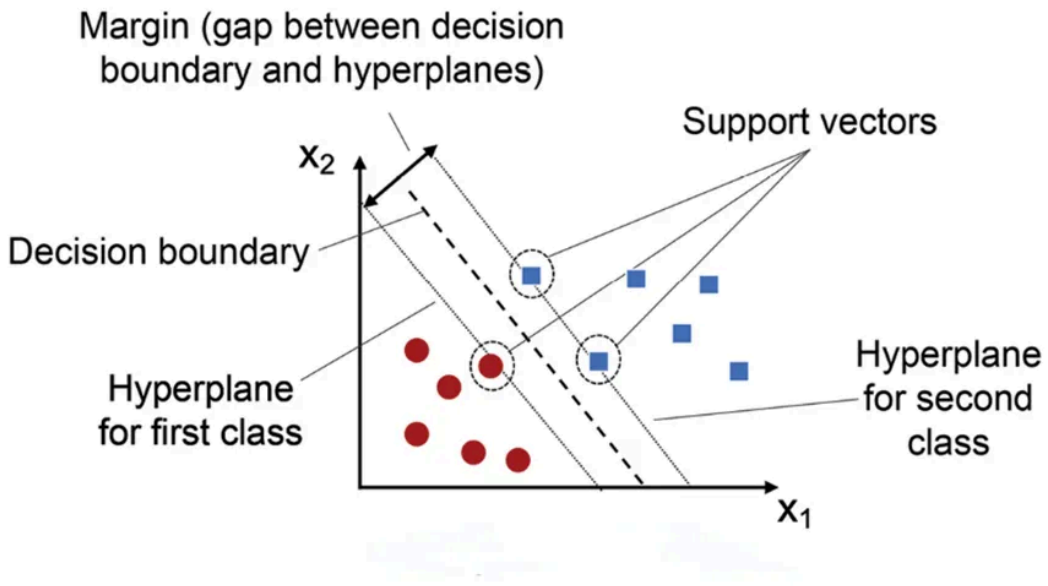
Support Vector Machines (SVM)

1. Introduction:

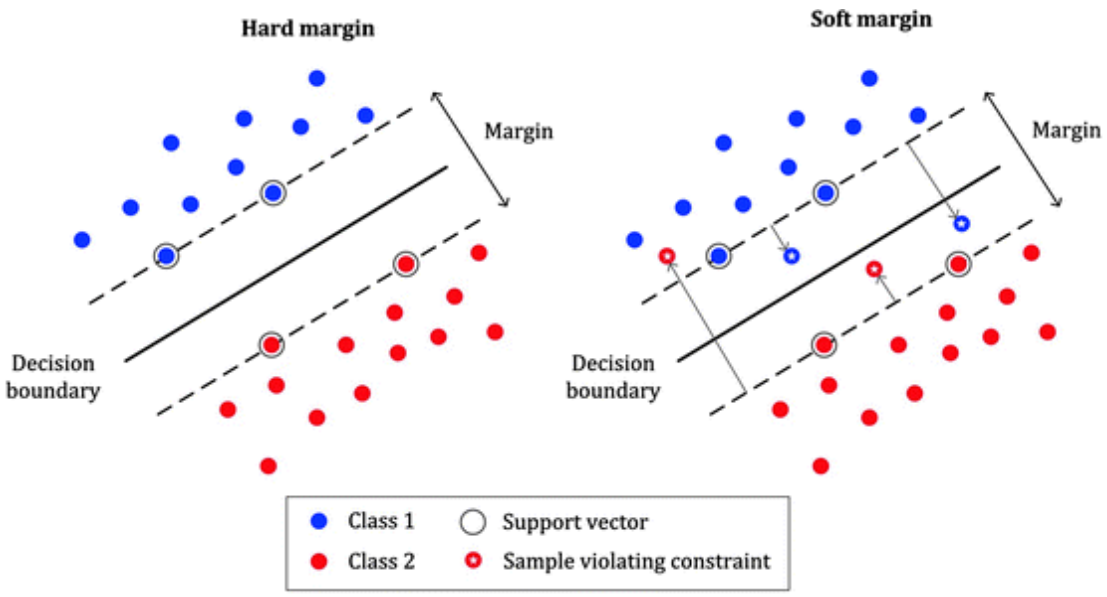
- **What it is:** SVM is a powerful and versatile supervised machine learning algorithm used primarily for **classification** tasks, although it can also be adapted for **regression** (Support Vector Regression - SVR).
- **Goal:** The fundamental idea behind SVM is to find an optimal **hyperplane** that best separates data points belonging to different classes in a high-dimensional space.

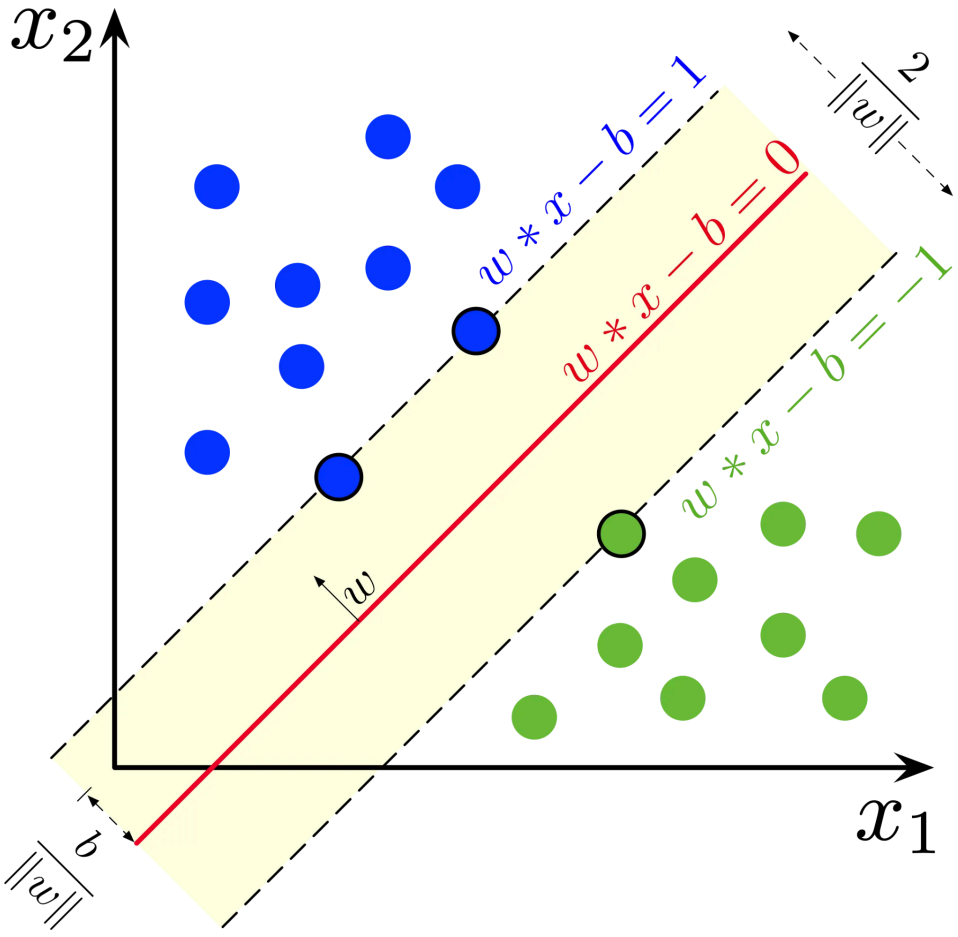
2. Core Concepts (Linear SVM):

- **Hyperplane:** In simple terms:
    - In a 2-dimensional space, a hyperplane is a **line**.
    - In a 3-dimensional space, a hyperplane is a **plane**.
    - In an N-dimensional space, it's an (N-1)-dimensional subspace.
    - This hyperplane acts as the **decision boundary** separating the classes.
- **Optimal Hyperplane:** For a given dataset, there might be many hyperplanes that can separate the classes. SVM aims to find the one that has the **maximum margin**.
- **Margin:** The margin is the distance between the hyperplane and the nearest data points (from *either* class). SVM tries to maximize this gap. A larger margin generally leads to better generalization performance (less overfitting).
- **Support Vectors:** These are the data points that lie closest to the hyperplane (on the edge of the margin). They are crucial because they are the *only* points that determine the position and orientation of the optimal hyperplane. If you remove other points (not support vectors), the hyperplane won't change. If you move a support vector, the hyperplane *will* likely change.

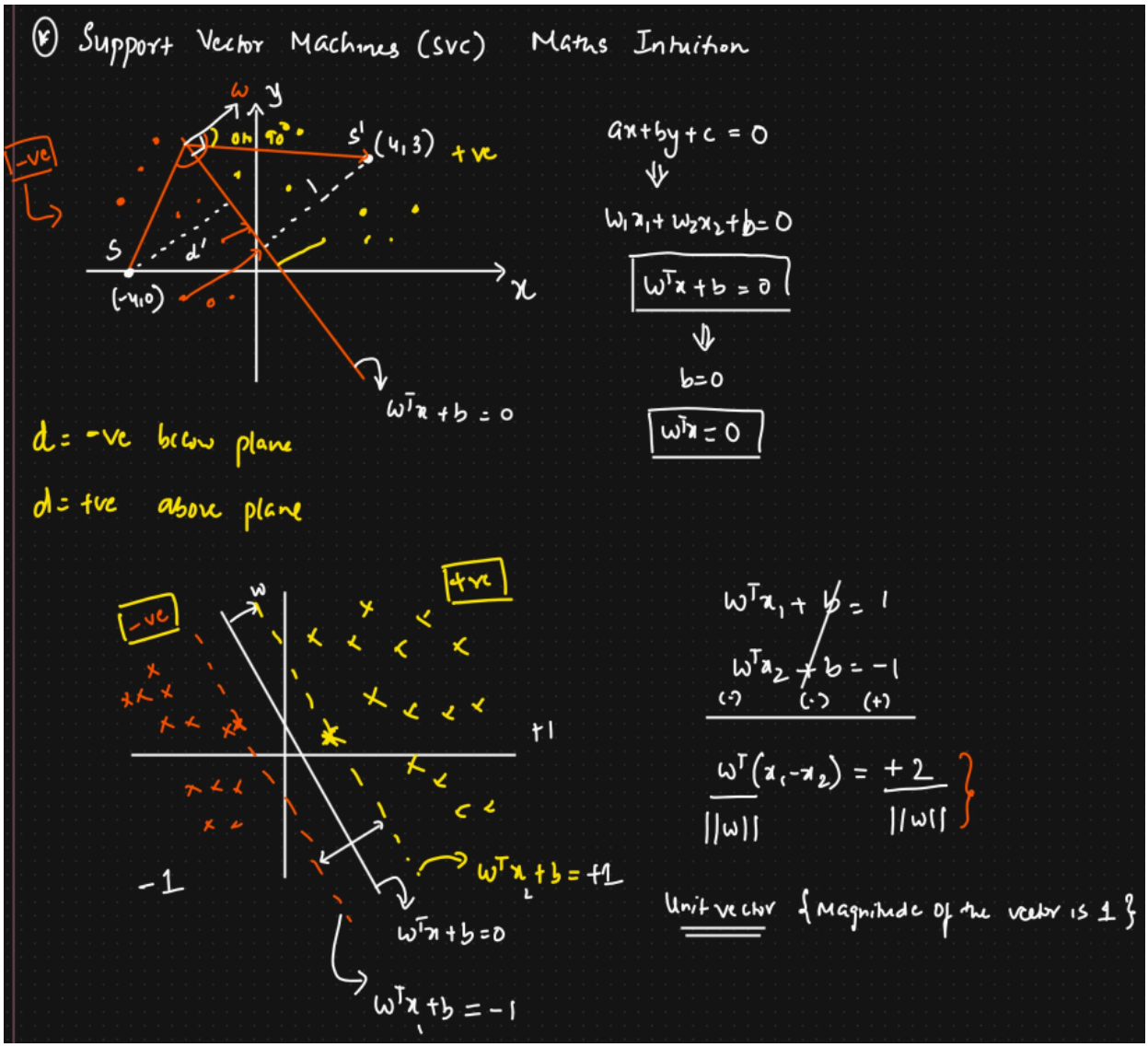


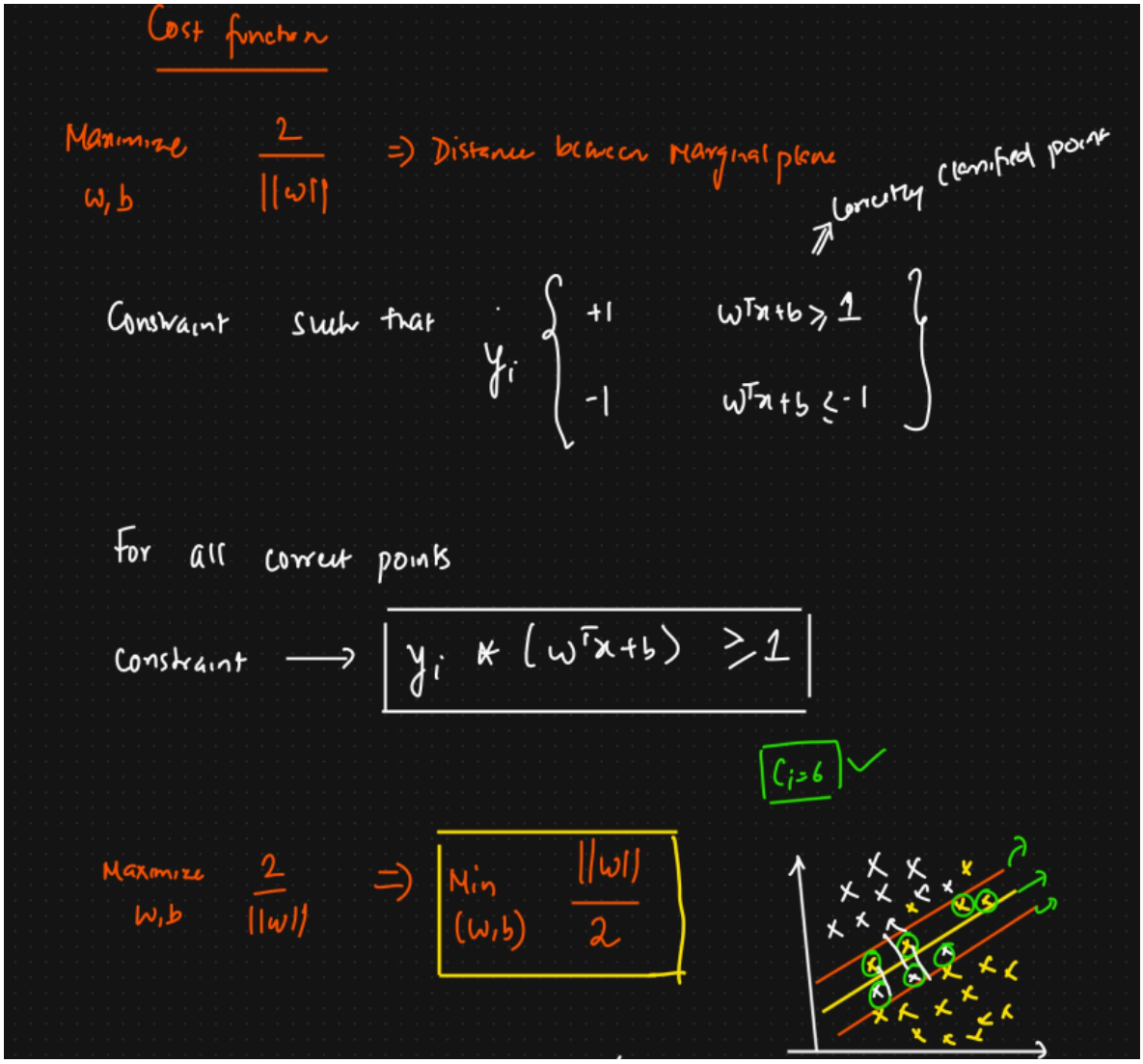
- **Hard Margin SVM:**
  - Assumes data is **perfectly linearly separable**.
  - Goal: Find a hyperplane that separates the classes with **zero misclassifications** and **no points inside the margin**.
  - Very sensitive to outliers.
  - Less practical for noisy, real-world data.
- **Soft Margin SVM:**
  - **Allows** for some **misclassifications** or points to fall inside the margin.
  - Handles data that is **not perfectly separable** (more realistic).
  - Uses a parameter (`C`) to control the **trade-off**:
    - Wider margin vs. Fewer classification errors.
  - More robust to outliers and noise.


Hard margin
Soft margin
Margin
Margin
Decision boundary
Decision boundary
Class 1
Class 2
Support vector
Sample violating constraint


x_2
x_1
$w * x - b = 1$
$w * x - b = 0$
$w * x - b = -1$
w
$\frac{2}{\|w\|}$
$\frac{b}{\|w\|}$

1. Hyperplane (in red): This is the decision boundary represented by the equation w · x — b = 0. It separates the two classes (green and blue points). The SVM algorithm searches for the hyperplane that best divides the data while maximizing the margin.
2. Support Vectors: These are the points closest to the hyperplane from both classes, highlighted on the boundary lines w · x — b = 1 and w · x — b = -1. In the figure, they are the points on the dashed lines representing the margin. Support vectors directly influence the positioning of the hyperplane.
3. Margin (in yellow): The margin is the distance between the support vectors and the hyperplane. The goal of SVM is to maximize this margin, ensuring that the hyperplane separates the classes as clearly as possible. In the figure, the margin is the region between the dashed lines w · x — b = 1 and w · x — b = -1.
4. Weight vector w: The arrow labeled w represents the weight vector perpendicular to the hyperplane. The direction of this vector indicates how the hyperplane is oriented, and its magnitude determines how steep the slope of the separation boundary is.





Distance from a Data Point to the Hyperplane:

$$\hat{y} = \begin{cases} 1 & : w^T x + b \geq 0 \\ 0 & : w^T x + b < 0 \end{cases}$$

Where $\hat{y}$ is the predicted label of a data point.

Optimization Problem for SVM

For a linearly separable dataset, the goal is to find the hyperplane that maximizes the margin between the two classes while ensuring that all data points are correctly classified. This leads to the following optimization problem:

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2$$

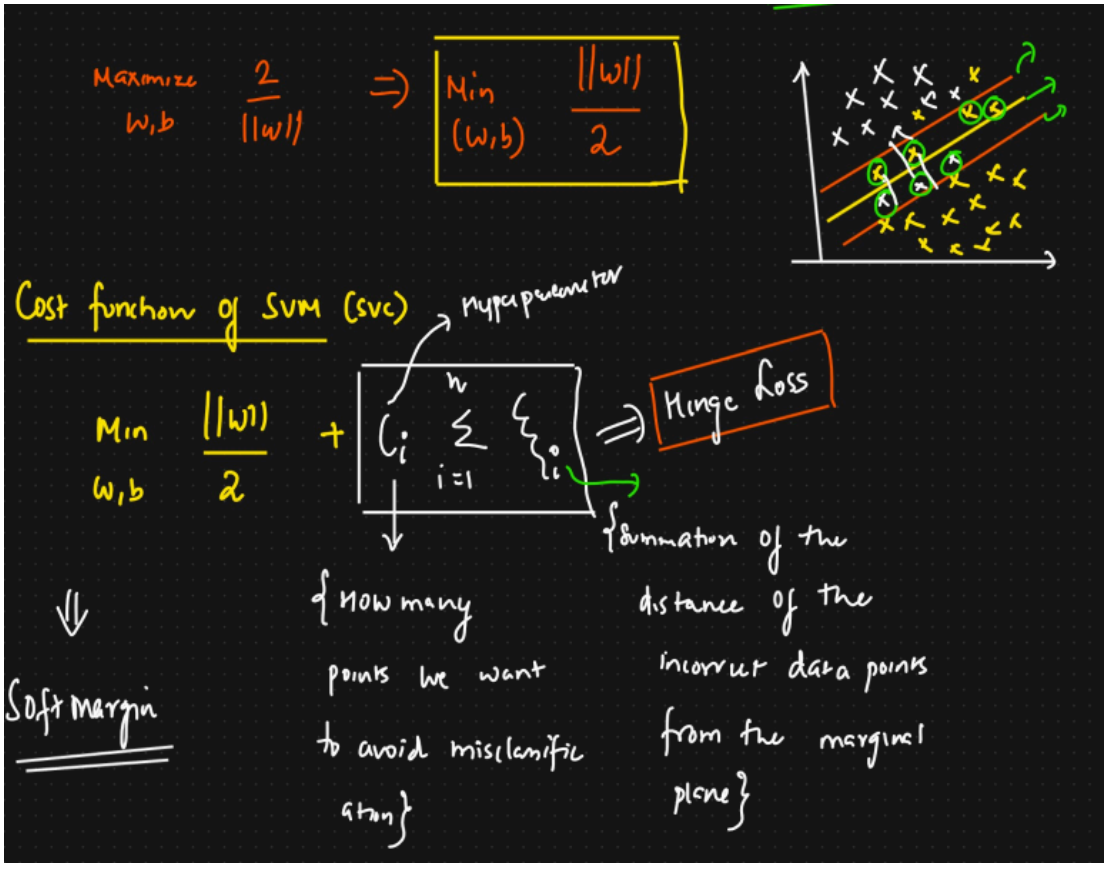
Subject to the constraint:

$$y_i(w^T x_i + b) \geq 1 \; for \; i = 1, 2, 3, \cdots, m$$

Where:

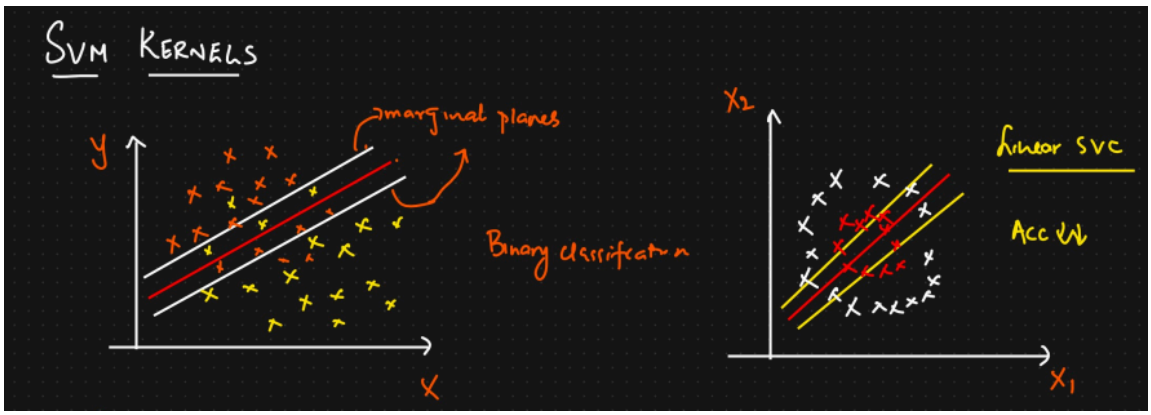
- y_i is the class label (+1 or -1) for each training instance.
- x_i is the feature vector for the i-th training instance.
- m is the total number of training instances.

The condition $y_i(w^T x_i + b) \geq 1$ ensures that each data point is correctly classified and lies outside the margin.

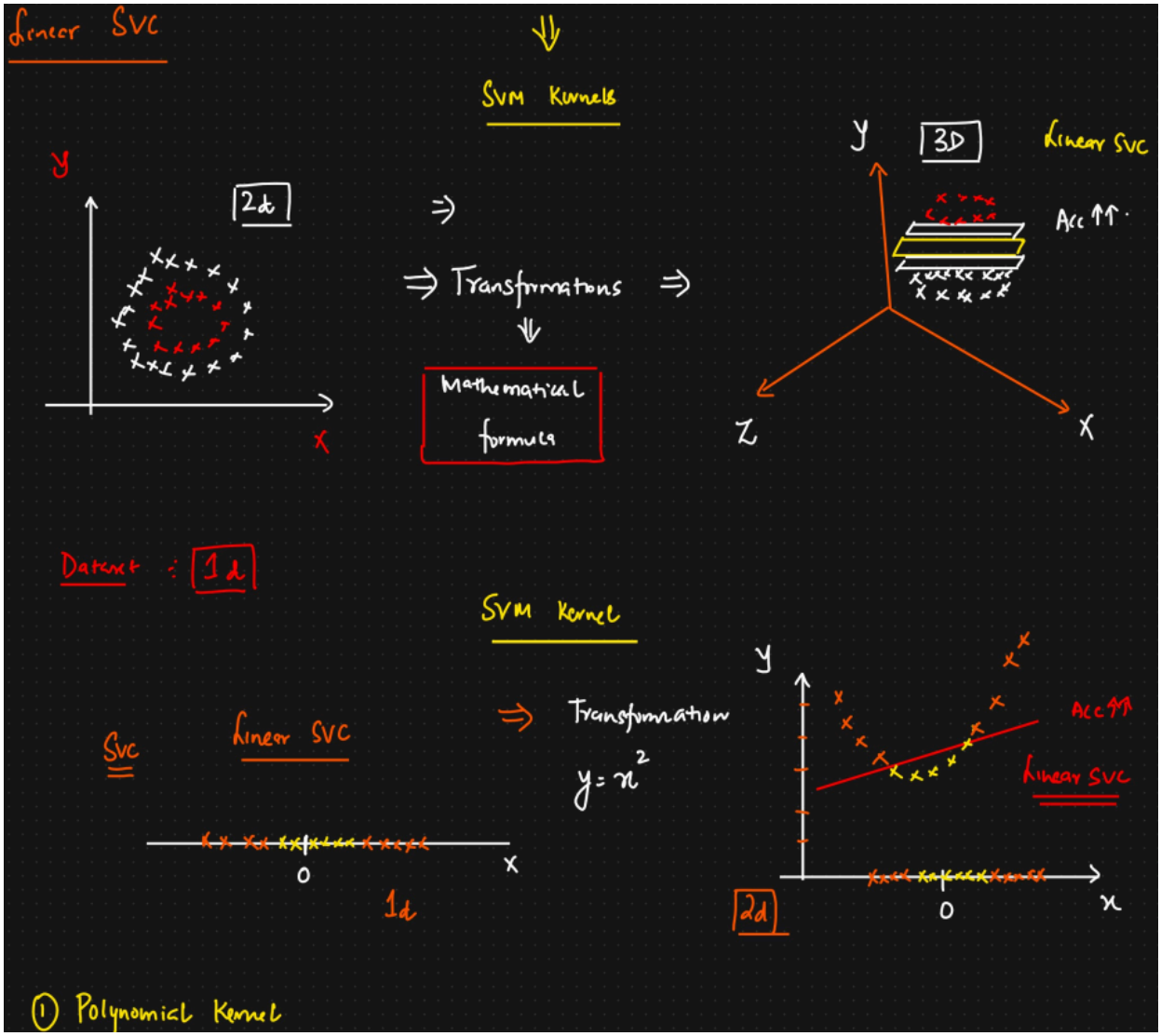


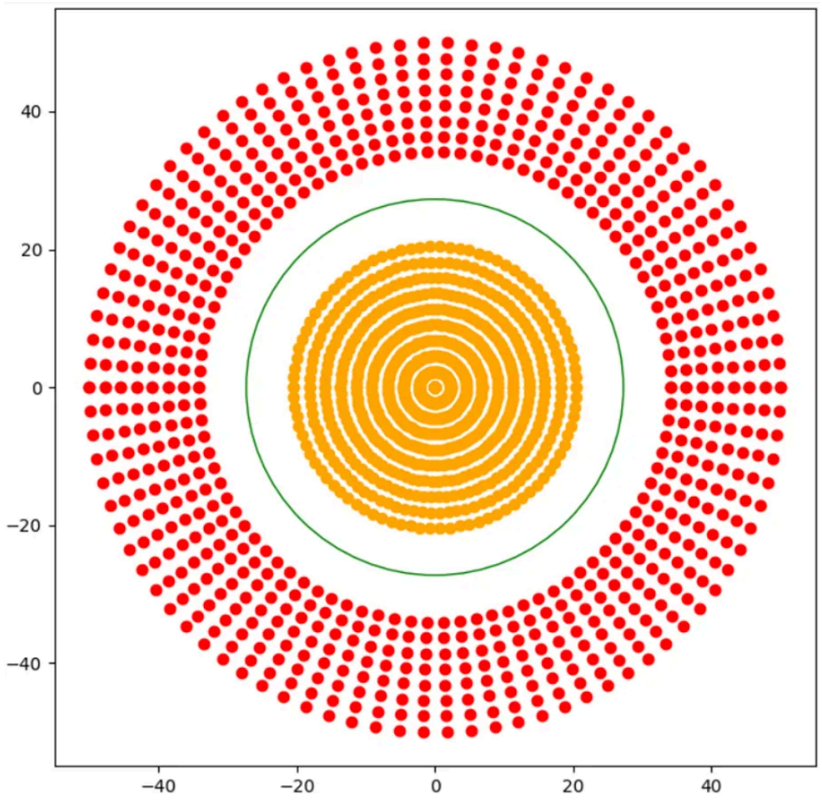
3. Handling Non-Linear Data: The Kernel Trick

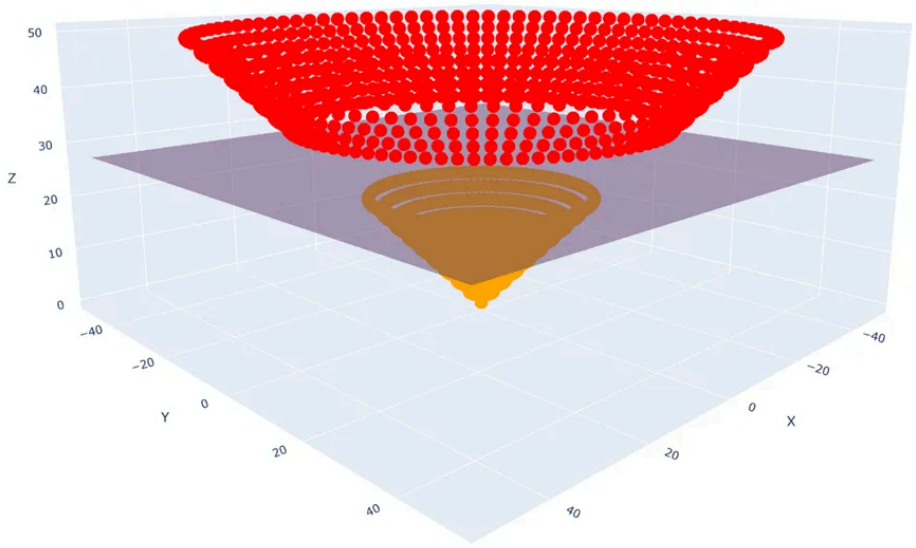
- **Problem:** Real-world data is often not **linearly separable** (i.e., you can't draw a single straight line/plane to separate the classes perfectly).



- **Solution: The Kernel Trick:** SVM uses a clever technique called the kernel trick to handle non-linear data.
    - **Idea:** It implicitly maps the data points from the original input space into a much higher-dimensional feature space *without actually computing the coordinates* in that new space.
    - **Benefit:** In this higher-dimensional space, the data often *becomes* linearly separable, allowing SVM to find a linear hyperplane there. This hyperplane, when mapped back to the original space, corresponds to a non-linear decision boundary.
    - **Efficiency:** Calculating coordinates in very high dimensions can be computationally expensive. Kernels compute the *relationship* (like dot products) between pairs of points in the higher-dimensional space directly from the original points, making the process feasible.
- **Common Kernel Functions:**
    - **Linear Kernel:** `K(x, y) = x^T y`. Used when the data is already linearly separable (equivalent to the basic linear SVM).
    - **Polynomial Kernel:** `K(x, y) = (\gamma x^T y + r)^d`. Introduces polynomial curves as decision boundaries. `d` is the degree, `\gamma` is a coefficient, `r` is a constant.
    - **Radial Basis Function (RBF) Kernel / Gaussian Kernel:** `K(x, y) = \exp(-\gamma ||x - y||^2)`. A very popular and powerful kernel, often a good default choice. It can create complex, localized decision boundaries. `\gamma` (gamma) controls the influence of a single training example – low gamma means far influence, high gamma means close influence.
    - **Sigmoid Kernel:** `K(x, y) = \tanh(\gamma x^T y + r)`. Can behave similarly to neural networks under certain parameters.


linear SVC
SVM Kernels
y
2d
x
Transformations
Mathematical formula
y
3D
linear SVC
Acc ↑↑.
z
x
Dataset : 1d
SVM Kernel
SVC
linear SVC
Transformation
$y = x^2$
0
x
1d
y
Acc ↑↑
linear SVC
2d
0
x
① Polynomial Kernel





4. Soft Margin Classification (Handling Overlap & Outliers)

- **Problem:** What if the data cannot be perfectly separated, even with kernels? Or what if we want a model that's less sensitive to outliers?
- **Solution: Soft Margin:** Instead of insisting on a perfectly separating hyperplane (hard margin), we allow some data points to be on the wrong side of the margin or even on the wrong side of the hyperplane (misclassified).

- **Hyperparameter C (Regularization Parameter):**
    - This parameter controls the trade-off between maximizing the margin and minimizing the classification error (number/degree of margin violations).
    - **Low `C`:** Wider margin, more tolerant of misclassifications (more regularization). Can lead to underfitting if too low.
    - **High `C`:** Narrower margin, tries hard to classify every point correctly (less regularization). Can lead to overfitting if too high, sensitive to outliers.
    - `C` is a crucial hyperparameter that needs tuning (e.g., using cross-validation).

5. Advantages of SVM:

- **Effective in high-dimensional spaces:** Works well even when the number of dimensions is greater than the number of samples.
- **Memory efficient:** Uses only a subset of training points (the support vectors) in the decision function.
- **Versatile:** Different Kernel functions can be specified for the decision function. Custom kernels can also be used.
- **Good generalization:** The margin maximization helps prevent overfitting, leading to good performance on unseen data (when tuned properly).

6. Disadvantages of SVM:

- **Computationally expensive:** Training time complexity can be high (O(n2) to O(n3) depending on implementation) for very large datasets.
- **Sensitive to kernel choice and hyperparameters:** Performance heavily depends on selecting the right kernel and tuning hyperparameters like `C` and `gamma` (for RBF). This often requires careful cross-validation.
- **Doesn't directly provide probability estimates:** Basic SVMs output a class label, not a probability. Techniques like Platt scaling can be used post-training to get probabilities, but they aren't intrinsic to the model.
- **Less interpretable:** The decision boundary, especially with kernels like RBF, can be hard to interpret compared to models like Decision Trees.

7. Applications:

SVMs are used in various fields, including:

- Image Classification
- Text Categorization (e.g., spam detection)
- Bioinformatics (e.g., protein classification, cancer classification)
- Handwriting Recognition
- Face Detection

Support Vector Regression (SVR)

1. Introduction:

- **What it is:** SVR is the regression counterpart to the Support Vector Machine (SVM) classification algorithm.
- **Goal:** Instead of finding a hyperplane to *separate* classes, SVR aims to find a function (hyperplane) that fits the data such that as many data points as possible lie *within* a certain margin or "tube" around the function.

2. Core Idea: The Epsilon-Insensitive Tube

- **Margin (ε - Epsilon):** SVR defines a margin of tolerance, denoted by ε (epsilon). This margin forms a "tube" around the regression line (hyperplane).
- **Objective:** The goal is to fit a function where the absolute error (residual) for most training data points is less than ε.
- **Epsilon-Insensitivity:** Crucially, any data points lying *inside* this tube (i.e., their prediction error is less than ε) do *not* contribute to the loss function. SVR is "insensitive" to errors smaller than ε. This differs from methods like Ordinary Least Squares (OLS) that penalize all errors.

3. Support Vectors in SVR:

- Similar to SVM, SVR relies on **support vectors**.
- These are the data points that lie **on the boundary** of the ε-tube or **outside** the tube.
- Points *inside* the ε-tube do not influence the final regression function. Only the support vectors dictate the position and shape of the regression line and the tube.

4. Soft Margin Regression (Handling Points Outside the Tube):

- **Problem:** Not all data points might fit perfectly within the ε-tube.
- **Solution:** SVR allows some points to fall outside the tube (soft margin concept).
- **Slack Variables:** Errors for points outside the tube are measured using slack variables.
- **Hyperparameter `C`:** This parameter controls the trade-off:
    - Between the "flatness" of the regression function (simplicity, wider effective tube) and the amount by which errors larger than ε are tolerated.
    - **Low `C`:** Tolerates more errors outside the tube, leads to a "flatter" (simpler) function.
    - **High `C`:** Penalizes errors outside the tube more heavily, tries to fit more points within or closer to the ε-tube boundary, potentially leading to a more complex function.

5. Kernels in SVR:

- Just like SVM classification, SVR can use the **kernel trick** (Linear, Polynomial, RBF, etc.).

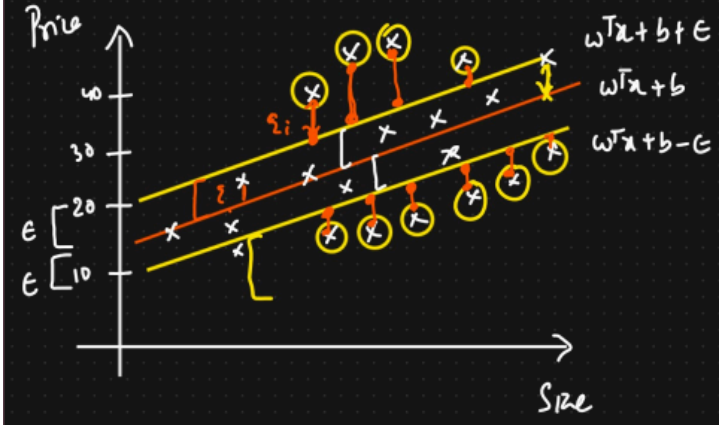
- This allows SVR to model **non-linear relationships** between the input features and the continuous target variable effectively.

6. Key Differences from SVM Classification:

- **Goal:** Predict continuous value (Regression) vs. Predict discrete class label (Classification).
- **Margin:** ε-insensitive tube around the function (SVR) vs. Separating boundary between classes (SVM).
- **Support Vectors:** Points on/outside the tube (SVR) vs. Points on the separating margin boundary (SVM).
- **Loss Function:** Ignores errors within ε (SVR) vs. Focuses on maximizing separation (SVM).

Support Vector Regression

ϵ : Marginal Error



Cost function

$$\underset{w,b}{\text{Min}} \quad \frac{\|w\|}{2} + C \sum_{i=1}^{n} \xi_i$$

Hyperparameter

$C \sum_{i=1}^{n} \xi_i$ → Hinge loss

Relationship

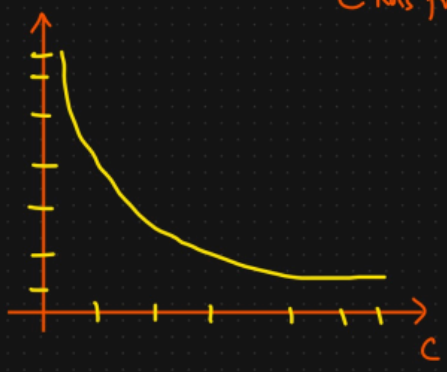
- C ↑↑
- loss function ↓↓

Constraint :

$$|y_i - w_i x_i| \leq \epsilon + \xi$$

⇓

loss function



ϵ → margin Error

ξ_i → Error above the margin