Introduction to Machine Learning and Support Vector Machines

What is Machine Learning?

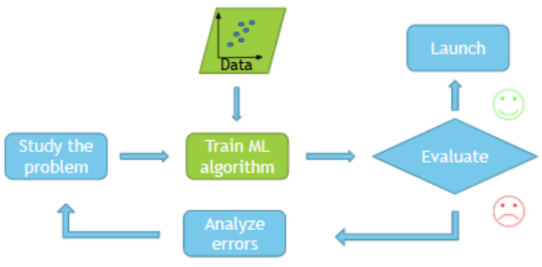
Machine Learning (ML) is a field of artificial intelligence that enables computers to learn from data and improve their performance without being explicitly programmed for every task.

KEY IDEA
Instead of writing specific rules, we provide data and let the algorithm discover patterns.

Example: Teaching a computer to recognize spam emails by showing it thousands of examples of spam and non-spam messages, rather than manually coding rules for every possible spam indicator.

Why Use Machine Learning?

ML vs Traditional Programming:

Traditional Programming: • Input Data + Rules (Code) → Output • We write explicit instructions for the computer to follow	 The traditional approach
Machine Learning: • Input Data + Output → Rules (Model) • The computer learns the rules from examples	 Machine Learning approach

---

When to Use ML?

Complex patterns: Problems where writing explicit rules is difficult or impossible

Example: Face recognition - too many variations in lighting, angles, expressions

Changing environments: Problems where rules need constant updating

Example: Stock market prediction - patterns change over time

Large-scale data: When you have massive amounts of data to analyze

Example: Recommendation systems (Netflix, Amazon) analyzing millions of user preferences

---

Exercise 1: Match the Following

Match each scenario to the appropriate approach:

Scenario	Approach
Calculate employee salary based on hours worked and fixed hourly rate	
Predict customer churn based on past behavior patterns	

Options: Traditional Programming, Machine Learning

Notes:

Categories of Machine Learning

1. Supervised Learning

The algorithm learns from labeled training data (input-output pairs).

Regression: Predicts continuous numerical values.

Examples:

- Predicting house prices based on size, location, number of rooms
- Forecasting temperature based on historical weather data
- Estimating a student's exam score based on study hours

Mathematical form: y = f(x) where y is a real number

Classification: Predicts discrete categories or classes.

Examples:

- Email spam detection (spam or not spam)
- Disease diagnosis (healthy, disease A, disease B)
- Image classification (cat, dog, bird)
- Credit approval (approve or reject)

Mathematical form: y = f(x) where y belongs to a set of classes $\{C_1, C_2, ..., C_K\}$

Notes:

2. Unsupervised Learning

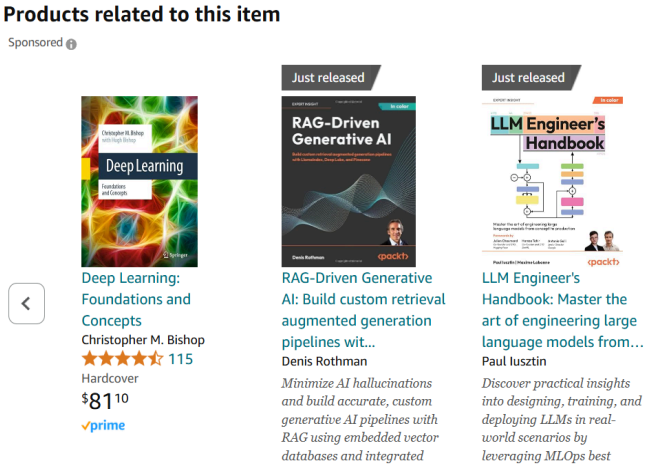
The algorithm finds patterns in unlabeled data without predefined outputs.

Clustering

Groups similar data points together.

Examples:

- Customer segmentation for marketing (grouping customers by buying behavior)
- Document organization (grouping news articles by topic)
- Gene sequence analysis (identifying related genes)



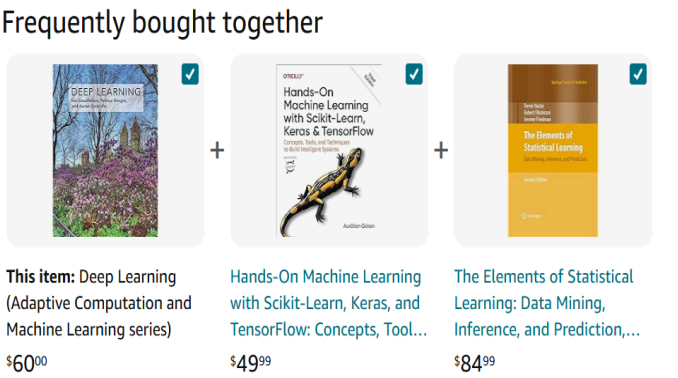
Association Analysis

Discovers relationships between variables in large datasets.

Examples:

- Market basket analysis: "Customers who buy bread also buy butter"
- Web usage mining: "Users who visit page A often visit page B"
- Medical records: Finding correlations between symptoms and conditions

Common rule form: If X then Y (e.g., If {milk, eggs} then {bread})



Notes:

3. Reinforcement Learning

The algorithm learns by interacting with an environment and receiving rewards or penalties.

Key Idea
Learning by trial and error - Agent takes actions → Gets feedback → Improves strategy

Key Components:

- **Agent:** The learner/decision maker
- **Environment:** What the agent interacts with
- **Actions:** Choices the agent can make
- **Rewards:** Feedback on actions (positive or negative)

Examples:

- Game playing (Chess, Go, video games) - agent learns winning strategies
- Robot navigation - robot learns to avoid obstacles and reach goals
- Self-driving cars - learning optimal driving behavior
- Resource management - optimizing server load balancing

Exercise 2: Match the Following

Match each scenario to the correct ML type:

Scenario	ML Type
Determining if a tumor is benign or malignant based on medical scans	
Finding groups of similar customers for targeted marketing campaigns	
Predicting the sale price of a used car based on mileage, age, and condition	
Discovering that customers who buy diapers often buy baby formula	
Teaching a robot to navigate a maze by rewarding it for reaching the exit	

Options: Regression, Classification, Clustering, Association Analysis, Reinforcement Learning

---

Notes:

Support Vector Machine (SVM)

Support Vector Machine is a powerful supervised learning algorithm used primarily for classification tasks. It works by finding the optimal boundary (hyperplane) that best separates different classes in the data.

Basic Idea
Find the best line/boundary that separates different classes with the maximum margin (distance) between them

Example: Imagine separating apples from oranges on a table. SVM finds the best "dividing line" that keeps the maximum distance from both the nearest apple and nearest orange.

Visual Concept:

- **Good separator:** A line that is far from both classes
- **Bad separator:** A line that is too close to one or both classes
- **Best separator (SVM):** The line with maximum margin from both classes

---

SVM as an Optimization Problem: SVM converts the classification problem into a mathematical optimization problem.

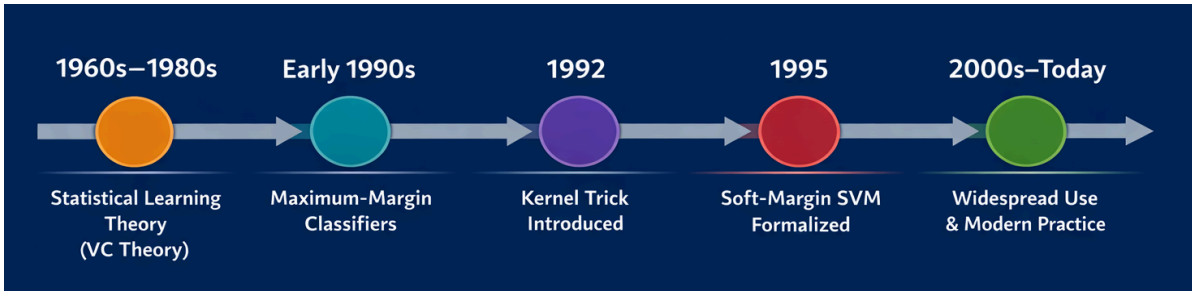
OPTIMIZATION OBJECTIVE
Maximize: The margin (distance between the decision boundary and nearest data points)
Subject to: All data points are correctly classified

Mathematical formulation:

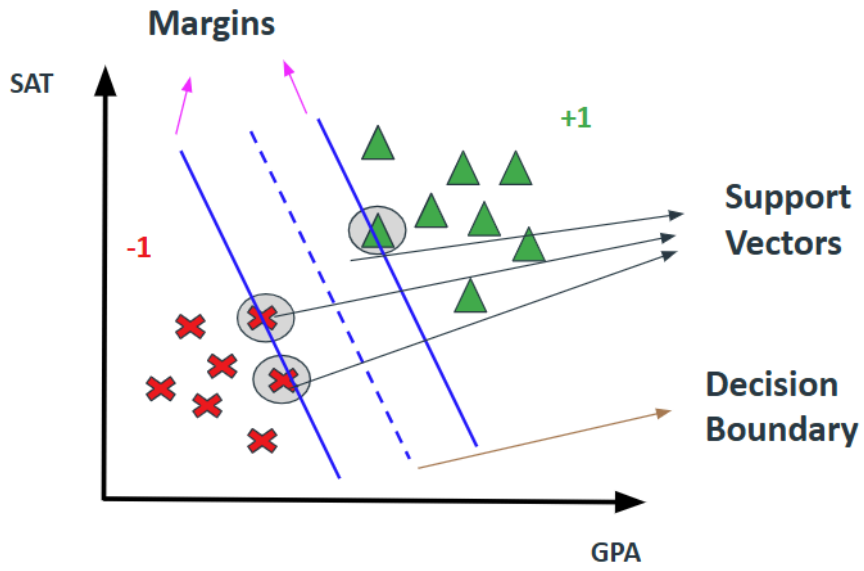
- Find the hyperplane: $w \cdot x + b = 0$
- Maximize: $2/\|w\|$ (the margin)
- Constraint: $y_i(w \cdot x_i + b) \geq 1$ for all data points

In simple terms: We want the widest possible "street" between classes, where the edges of the street touch the nearest points from each class.

History and Development of SVM:



Key Terminologies

Term	Definition
Hyperplane	The decision boundary that separates classes (a line in 2D, plane in 3D, etc.)
Support Vectors	The data points closest to the hyperplane; these define the margin
Margin	The distance between the hyperplane and the nearest support vectors
Kernel	A function that transforms data into higher dimensions for non-linear separation
	

Important notes:

- Only support vectors matter for defining the boundary
- Removing other points doesn't change the hyperplane
- The wider the margin, the better the generalization

Example: In a 2D space with two classes:

- Hyperplane = the dividing line
- Support vectors = the closest points on either side of the line
- Margin = the width of the gap between the two classes

Hard Margin vs Soft Margin SVM

HARD MARGIN SVM: Assumes data is perfectly linearly separable with NO misclassifications allowed

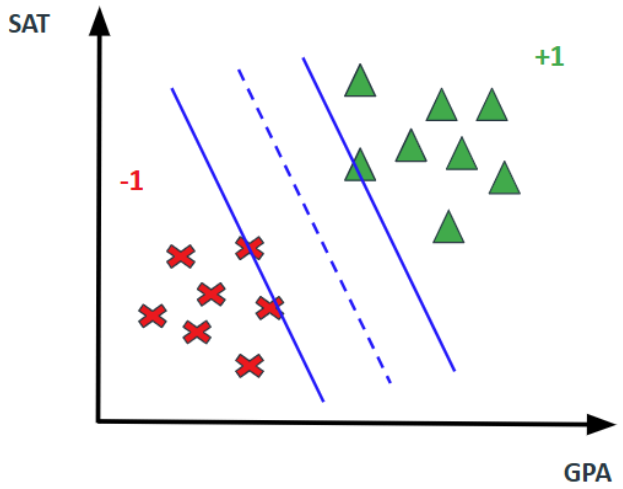
Characteristics:

- Requires all data points to be on the correct side of the margin
- No tolerance for errors or outliers
- Works only when classes are completely separable

Limitations:

- Very sensitive to outliers
- May not find a solution if data has even slight overlap
- Can lead to overfitting

Example: Like requiring every student to score above 90% - no exceptions allowed, which might be too strict.



SOFT MARGIN SVM: Allows some misclassifications and violations of the margin for more flexibility

Characteristics:

- Introduces slack variables (ξ) to allow some errors
- Balances margin maximization with misclassification penalty
- More robust to outliers and noise

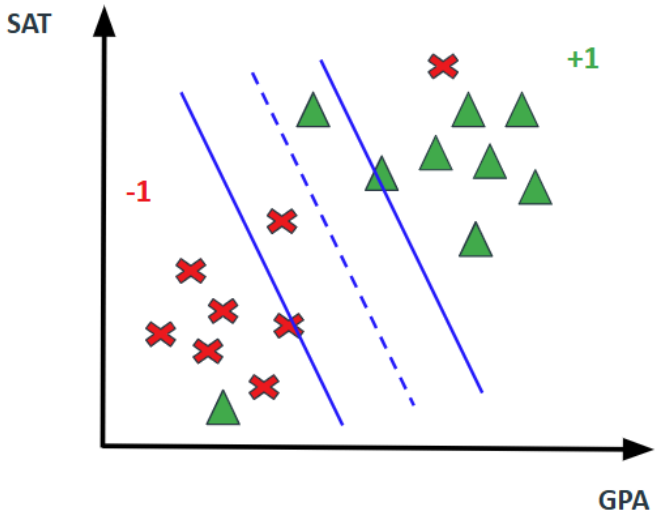
The Trade-off (C parameter):

- **Large C:** Fewer misclassifications allowed (stricter, like hard margin)
- **Small C:** More misclassifications tolerated (more flexible)

Modified Optimization:

- Minimize: $(1/2)\|w\|^2 + C\sum\xi\_i$
- Where C controls the penalty for misclassifications

Example: Like allowing some students to score 85-89% and still pass - more realistic and flexible.



When to Use Which?

Condition	Recommended Approach
Data is perfectly separable, no noise	Hard Margin SVM
Data has outliers or some overlap	Soft Margin SVM
Real-world applications	Soft Margin SVM (almost always)

PRACTICAL TIP
In practice, soft margin SVM is almost always preferred as real-world data is rarely perfectly separable

---

The Kernel Trick

Real-world data is often not linearly separable in its original space. The kernel trick is a clever mathematical technique that allows SVM to handle non-linear decision boundaries.

THE KERNEL TRICK

Transform data into a higher-dimensional space where it becomes linearly separable, WITHOUT explicitly computing the transformation

The Problem: Non-Linear Separability

In the original space, some data cannot be separated by a straight line (or hyperplane).

Example: Consider two classes arranged in concentric circles:

- Inner circle = Class A (e.g., red points)
- Outer circle = Class B (e.g., blue points)
- No straight line can separate them!

The Solution: Higher Dimensional Projection

Key Idea: Project the data into a higher-dimensional space where a linear separator (hyperplane) exists.

Simple 2D to 3D Example:

- Original space (2D): Points with coordinates (x_1, x_2)
- Transform to 3D: Add a new dimension $z = x_1^2 + x_2^2$
- In 3D, a plane can now separate the circles!

Visual Analogy: Imagine ants on a paper (2D) that can't be separated by a line. Lift some ants up (add a 3rd dimension), and now you can slide a flat sheet between them to separate them.

How the Kernel Trick Works?

THE MAGIC OF KERNELS
Instead of explicitly transforming data (expensive!), we use a kernel function that computes similarity in the higher dimension WITHOUT actually going there

Mathematical Insight:

- Direct approach: $\varphi(x)$ transforms to high dimension, then compute $\varphi(x_1) \cdot \varphi(x_2)$
- Kernel trick: $K(x_1, x_2)$ directly computes the dot product without transformation
- Result: Same answer, much faster computation!

Why This Matters:

- Transforming to very high dimensions (even infinite!) is computationally expensive
- Kernel functions compute the result efficiently
- We get non-linear decision boundaries in the original space

---

Practical Example

Problem: Classify emails as spam or not spam

Original Features (2D):

- x_1 = number of exclamation marks
- x_2 = number of capital letters

Issue: Spam and non-spam emails might not be linearly separable in this 2D space.

Solution with Kernel:

1. Use kernel to implicitly map to higher dimensions
2. SVM finds a hyperplane in this higher-dimensional space
3. Back in 2D, this appears as a curved (non-linear) decision boundary
4. Successfully separates spam from non-spam!

KEY TAKEAWAY

Kernels let SVM handle complex, non-linear patterns by implicitly working in higher dimensions while keeping computation efficient

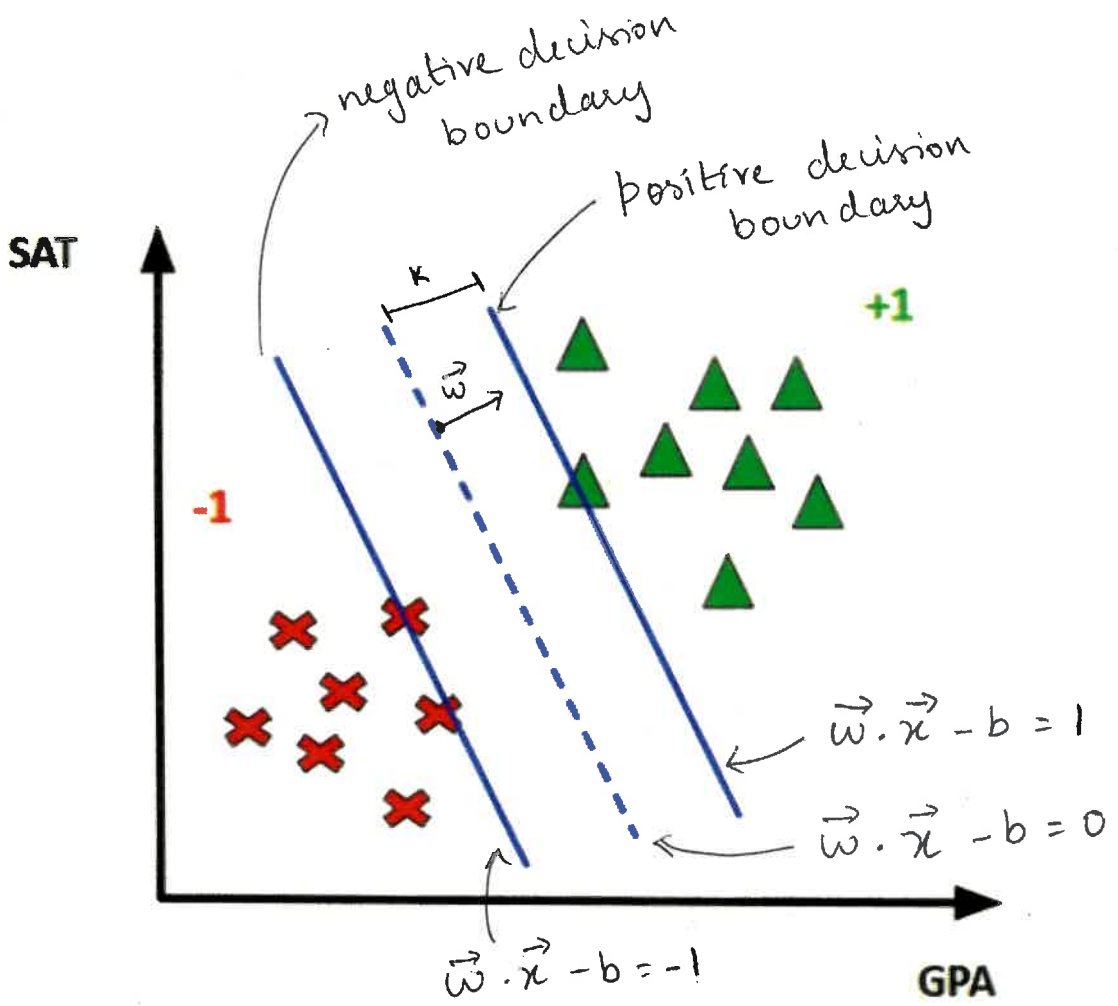
---

Exercise 3: True or False

Statement	T/F
A. Support vectors are all the data points in the dataset	
B. A larger margin generally leads to better generalization	
C. The kernel trick explicitly computes coordinates in higher dimensions	
D. Soft margin SVM is more practical for real-world data than hard margin	

Training data

$$\begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \cdots \\ \vec{x}_n \end{bmatrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$



Let $\vec{x}_0$ be any point on the decision boundary

$$\Rightarrow \quad \vec{w} \cdot \vec{x}_0 - b = 0 \quad - (1)$$

We want to walk from $\vec{x}_0$ in the direction of w (the perpendicular direction) until we reach the positive margin boundary.

Lets walk a distance of k units in the direction of the unit vector, $\frac{w}{\|w\|}$

$\therefore$ the new position after walking k units:

$$\vec{x}_1 = \vec{x}_0 + k \frac{w}{\|w\|} \quad - (2)$$

Since x_1 is on the positive margin boundary,

$$\vec{w} \cdot \vec{x}_1 - b = 1 \quad - (3)$$

substitute (2) in (3)

$$\vec{w} \cdot \left(\vec{x}_0 + k \frac{w}{\|w\|} \right) - b = 1$$

$$\vec{w} \cdot \vec{x}_0 + k \frac{w \cdot w}{\|w\|} - b = 1$$

$$\left(\vec{w} \cdot \vec{x}_0 - b \right) + k \frac{w \cdot w}{\|w\|} = 1 \quad - (4)$$

substitute (1) in (4), $\quad w \cdot w = \|w\|^2$

$$0 + k \frac{\|w\|^2}{\|w\|} = 1$$

$$\Rightarrow \quad \boxed{k = \frac{1}{\|\vec{w}\|}}$$

$\Rightarrow$

$\therefore$ The margin size is $\frac{2}{\|\vec{w}\|}$

So, we have to maximize the margin

$\Rightarrow$ minimizing $\|\vec{w}\|$

Constraints:

1. We need every data point be correctly classified
2. Every data point be outside or on the margin

ie., for positive classes,

$$y_i = +1 \quad , \quad \vec{w} \cdot \vec{x}_i - b \geq +1$$

for negative classes

$$y_i = -1 \quad , \quad \vec{w} \cdot \vec{x}_i - b \leq -1$$

we combine both constraints by multiplying y_i

$$y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1$$

$\therefore$ minimize $\|w\|$

subject to $y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall \; i = 1, 2 \ldots n$