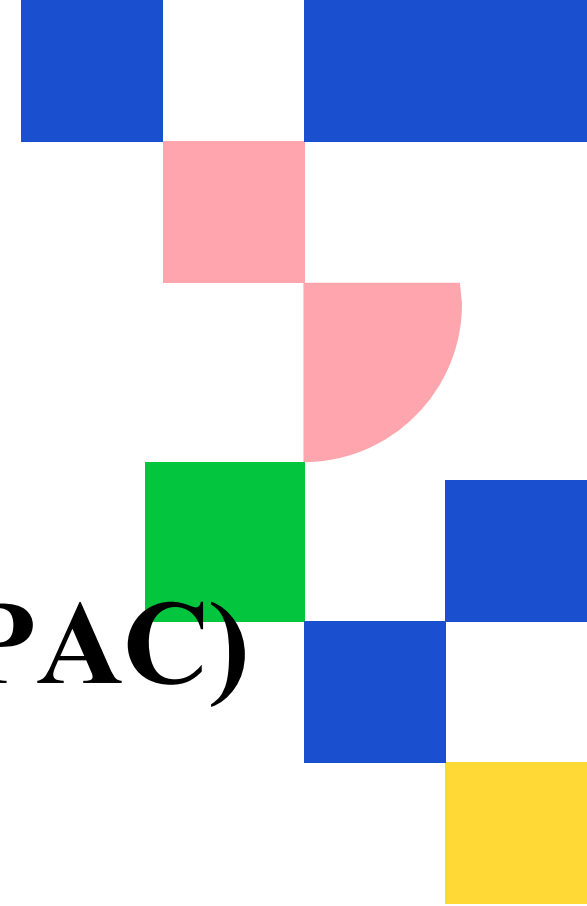


Probably Approximately Correct (PAC) Learning



PAC Learning

- **Definition:** PAC learning is a **theoretical framework** in machine learning that helps us understand whether an algorithm can **learn a concept correctly and efficiently** from data, with **high probability**.
- **Introduced by Leslie Valiant in 1984** through his seminal paper *"A Theory of the Learnable"*

PAC Learning

It addresses key questions about algorithms

- **What can be learned efficiently,**
- **What is inherently difficult to learn, and**
- **How many examples are required for successful learning.**

Under what conditions can a learning algorithm guarantee finding a **hypothesis** that is "**approximately correct**" with "**high probability**" when presented with a "**reasonable**" number of training examples?

PAC Learning – key idea

Approximately Correct refers to the **accuracy**, meaning the **error** of the hypothesis is at most a small value, denoted by **ϵ (epsilon) > 0** .

- **Empirical Error** - The error on the **training data**.
- **Generalization Error** - The actual error the hypothesis makes on **unseen data**.
- So, The true generalization **error to be $\leq \epsilon$ (at most)**.
- **Accuracy = $1 - \epsilon$**

PAC Learning – key idea

Probably Correct refers to the **confidence**, implying that this approximation holds with a high probability, **at least $1 - \delta(\text{delta})$** .

- Since the training data is drawn randomly, there's always a chance that the specific sample we get is misleading.
- So, we want **$P(\text{error}(h) \leq \epsilon) \geq 1 - \delta$** .
- δ is the **probability of failure** is at most.
- **Confidence = $1 - \delta$**

PAC Learning – key terms

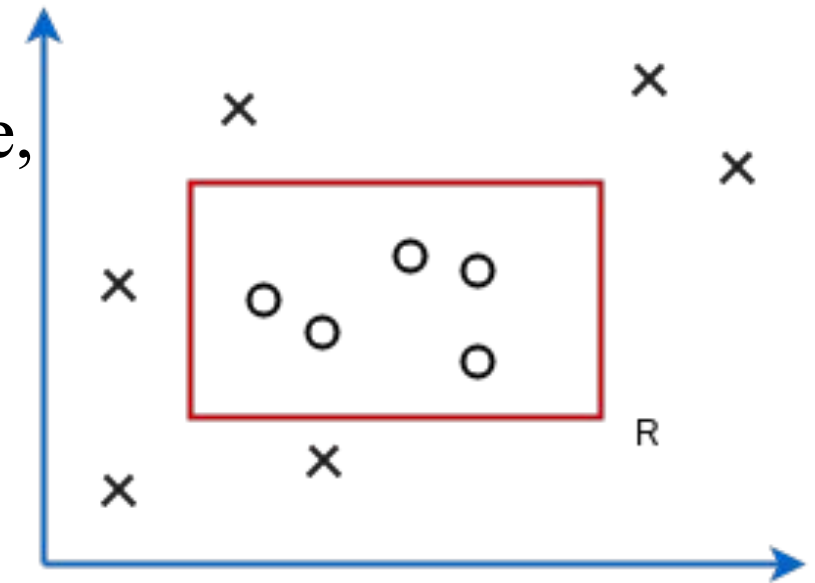
Symbol	Meaning
ε	Accuracy level (small error)
δ	Confidence level (small risk)
H	Hypothesis set
h	Chosen hypothesis
c	True concept
D	Unknown data distribution

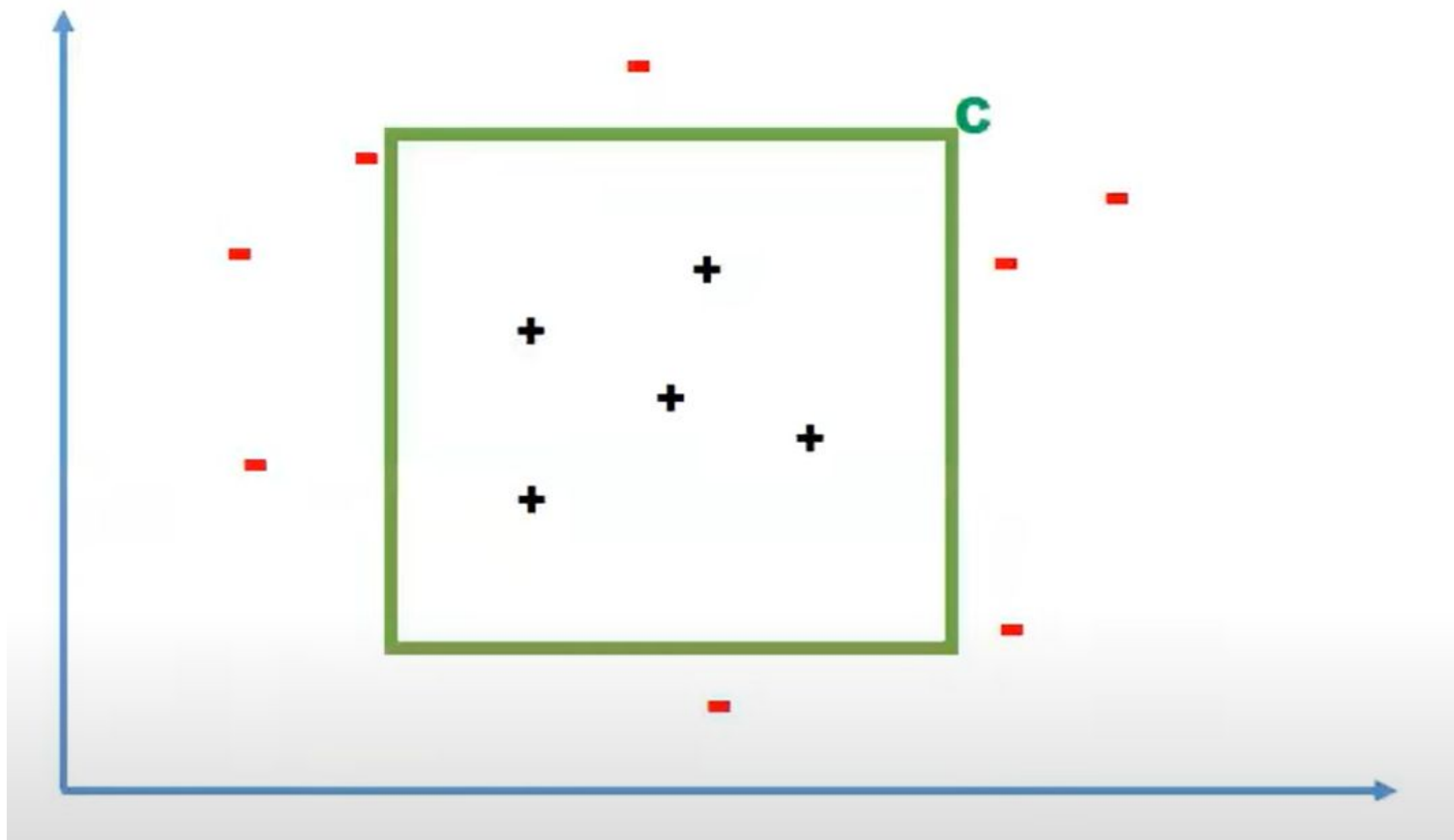
PAC Learning – Examples of Learnability

Axis-Aligned Rectangles:

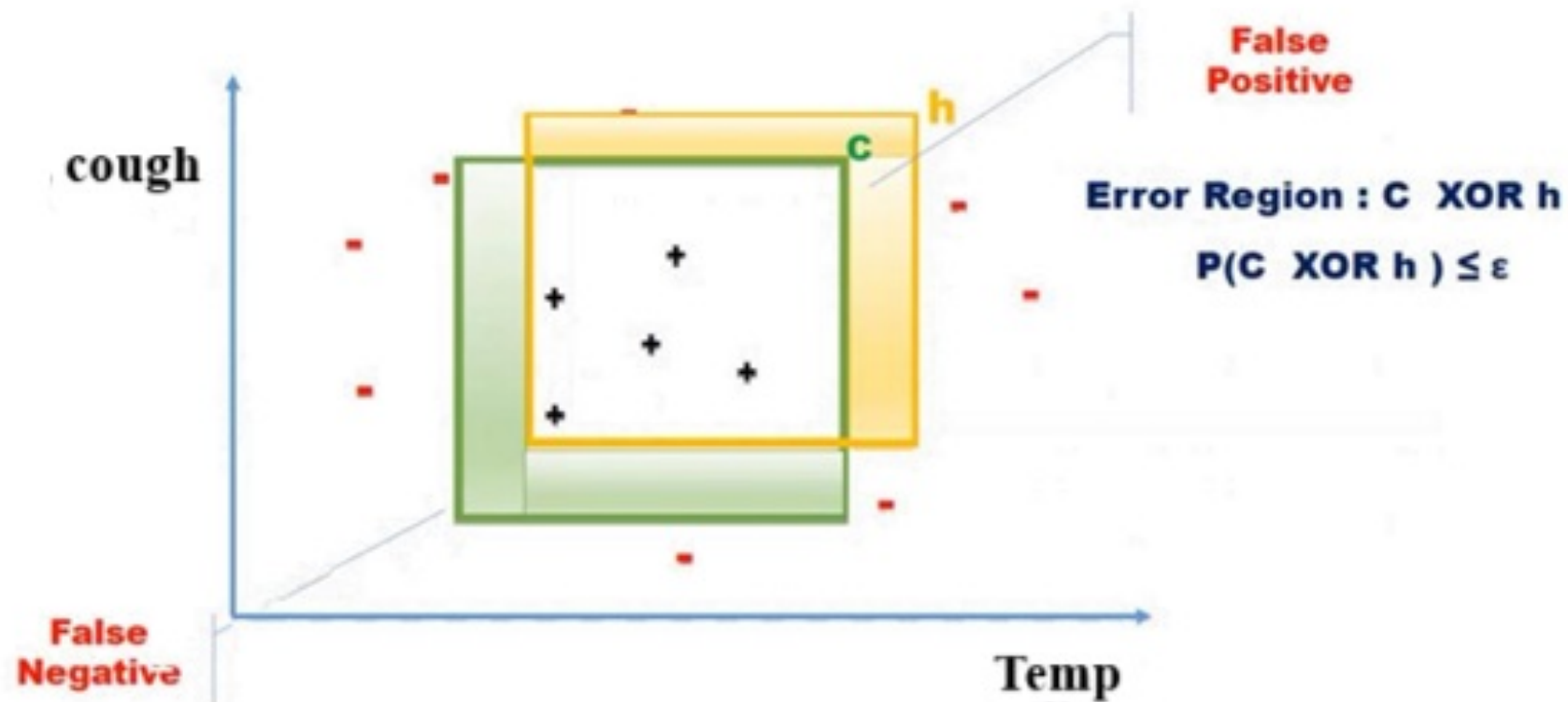
A simple algorithm involves returning the **tightest axis-aligned rectangle** containing the positive examples.

This infinite concept class is indeed PAC-learnable, with sample complexity $O((1/\epsilon) \log(1/\delta))$

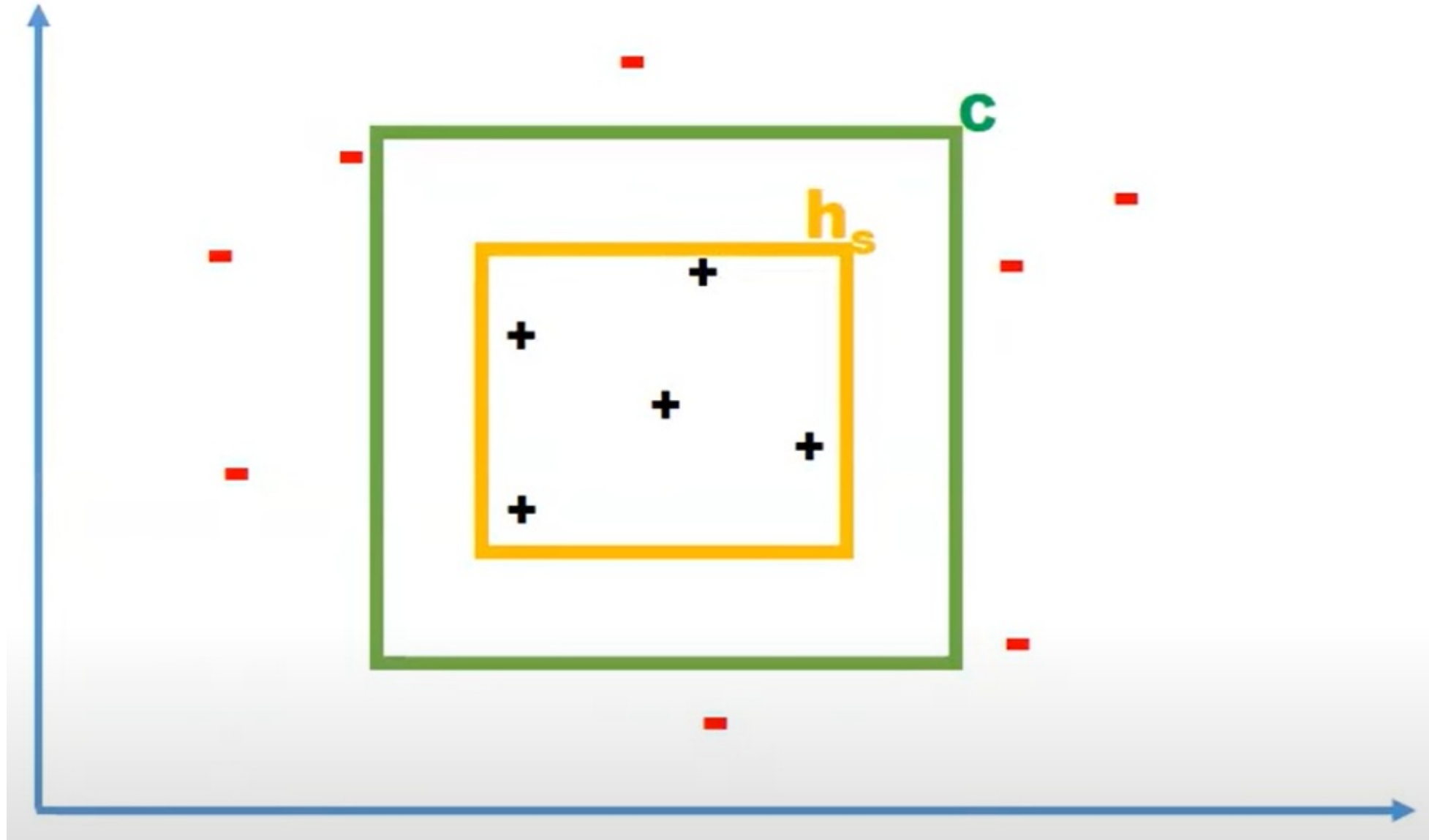


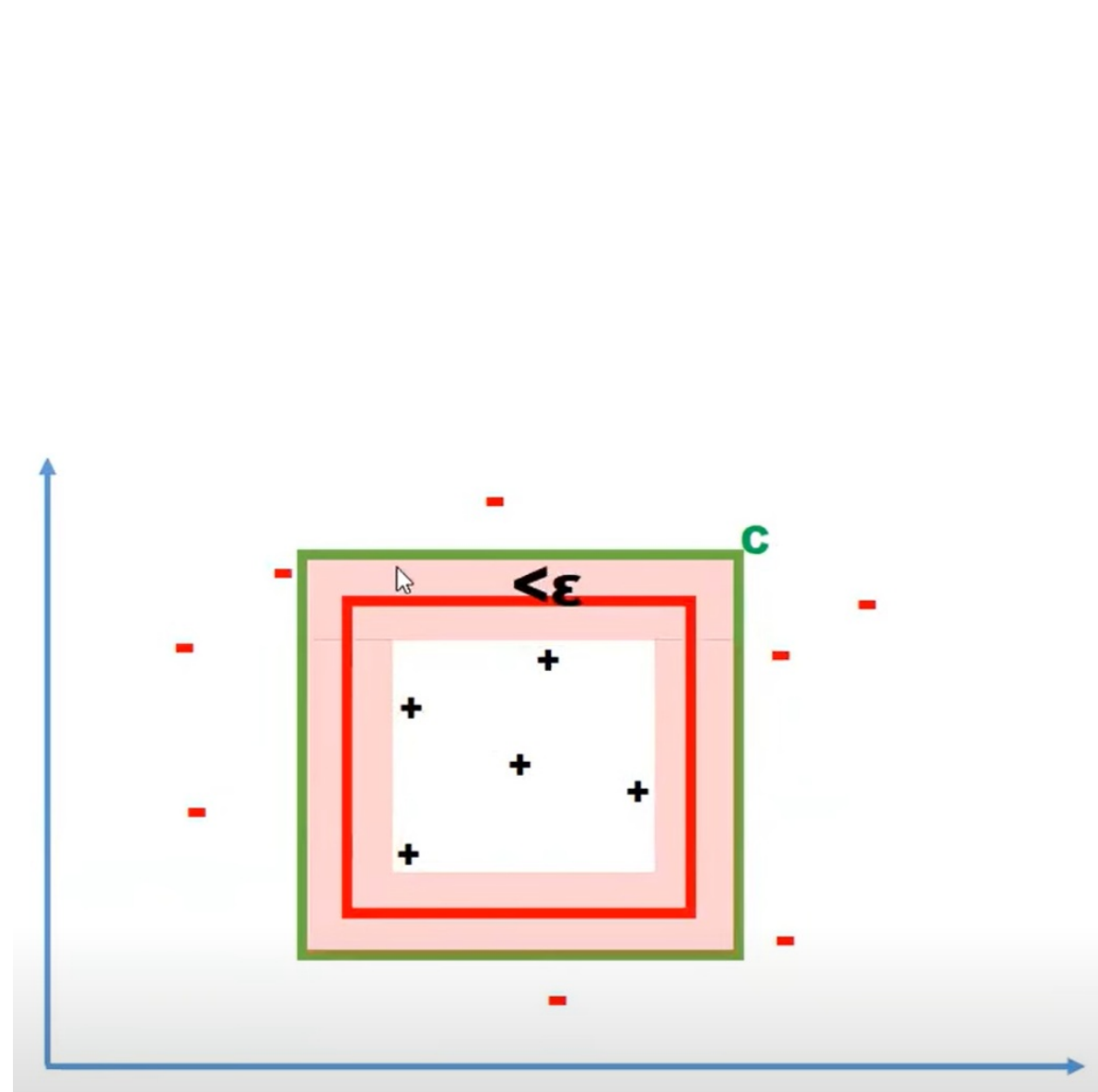
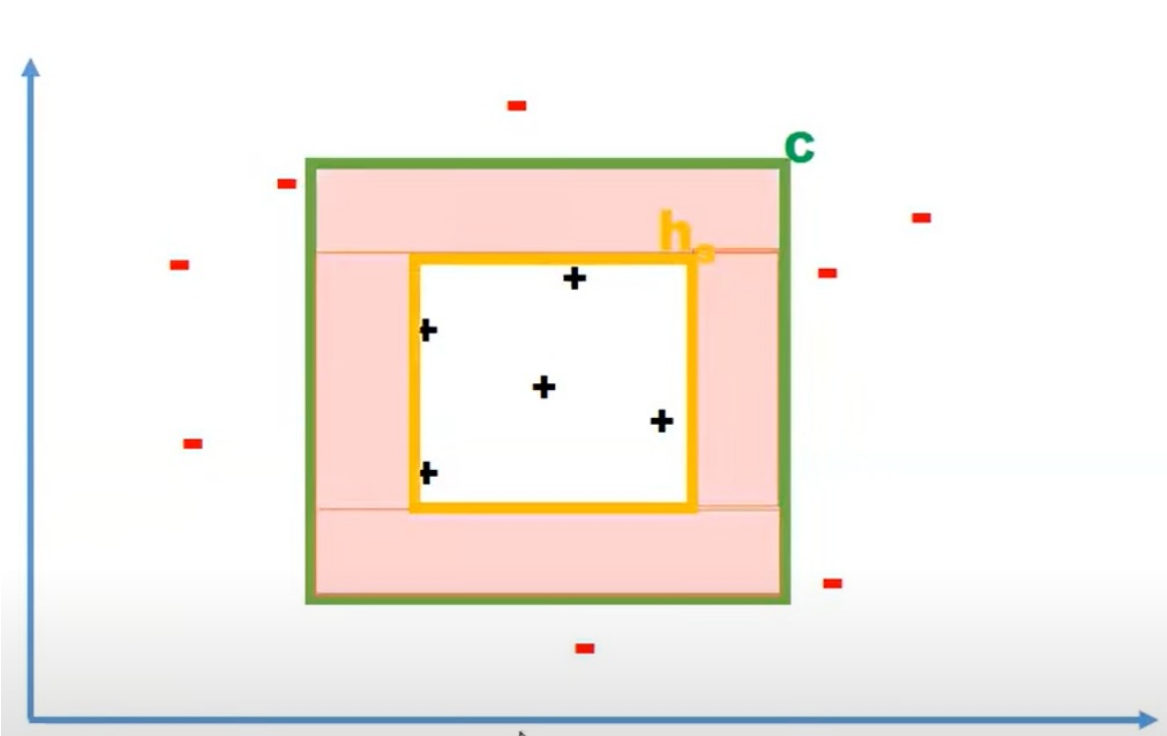


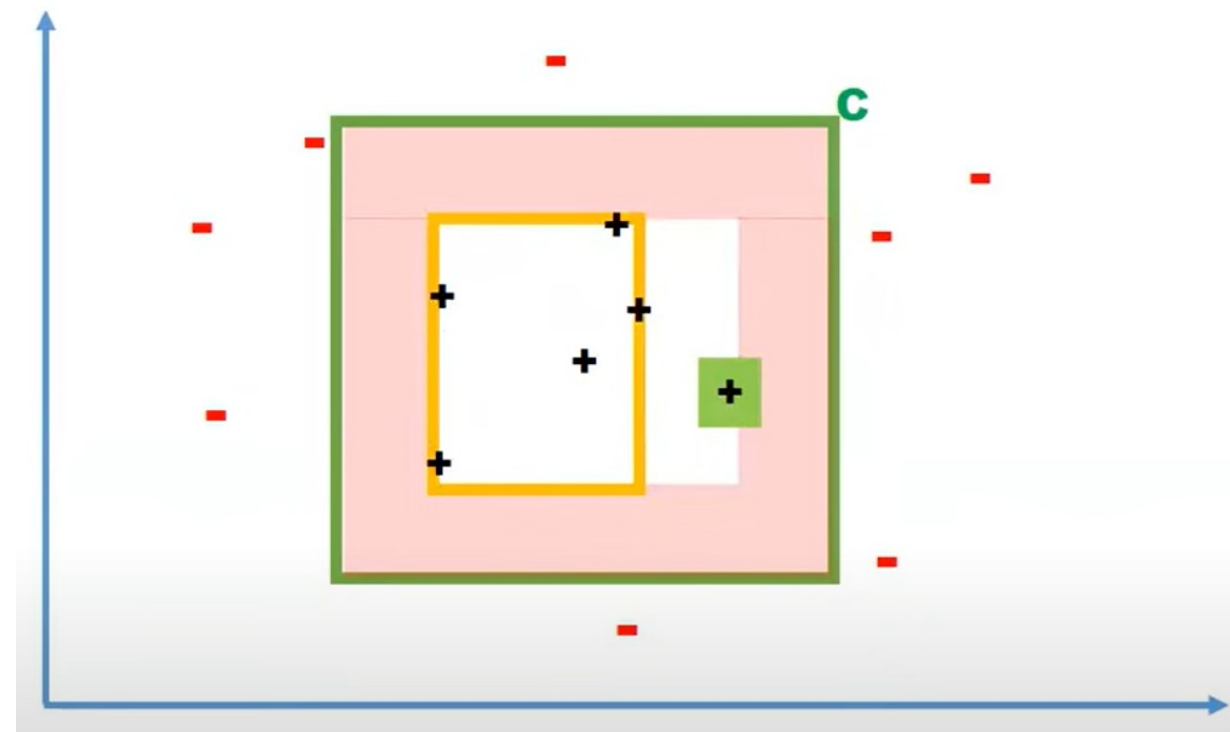
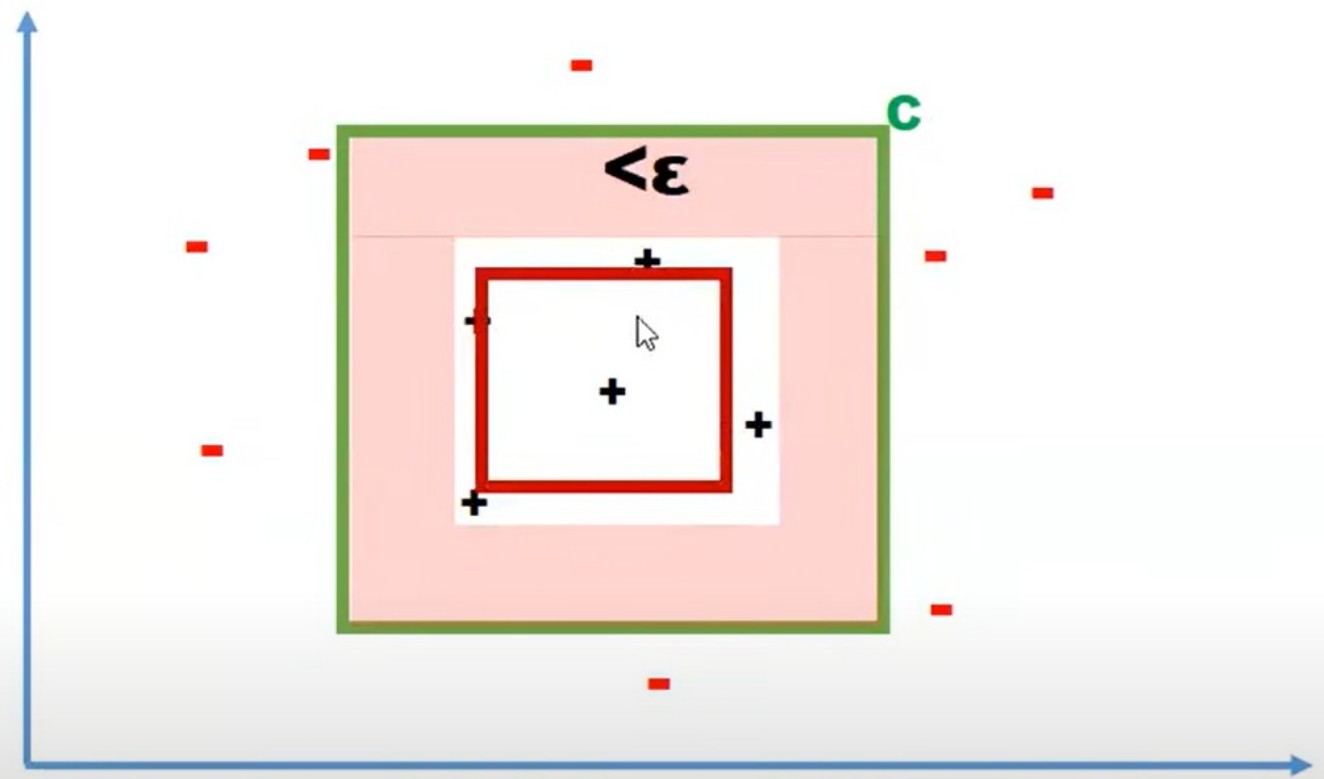
Probably Approximately Correct (PAC) learning



A simple algorithm involves returning the **tightest axis-aligned rectangle** containing the positive examples.







Error Region = Sum of four rectangular strips $< \epsilon$

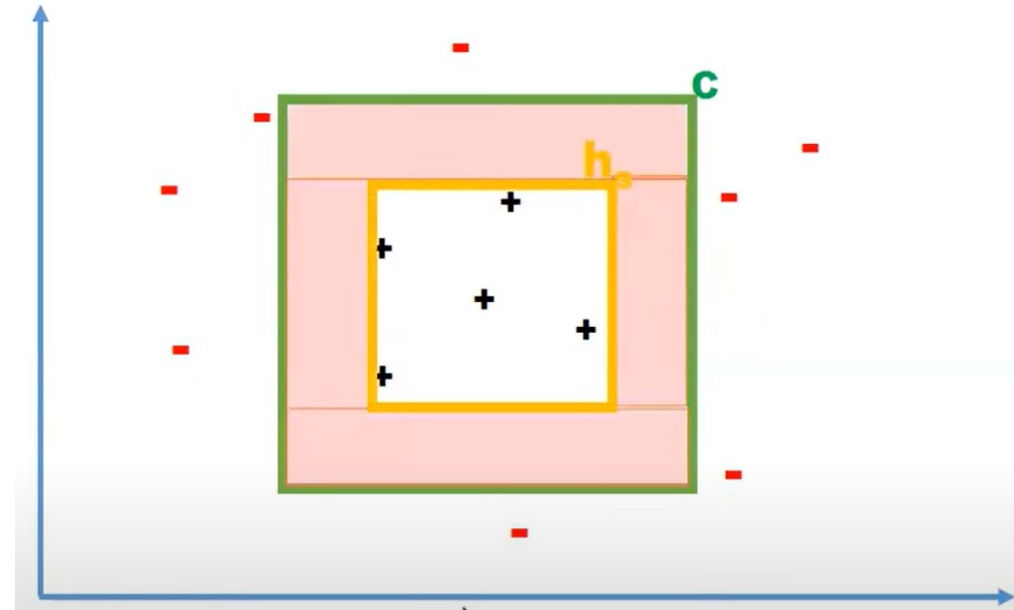
Each strip is at most $\epsilon / 4$

Probability of a positive example falling in any one of the strip (error region) = $\epsilon / 4$

Probability that a randomly drawn positive example misses a strip = $1 - \epsilon / 4$

$P(\text{m instances miss a strip}) = (1 - \epsilon / 4)^m$

$P(\text{m instances miss any strip}) < 4(1 - \epsilon / 4)^m$

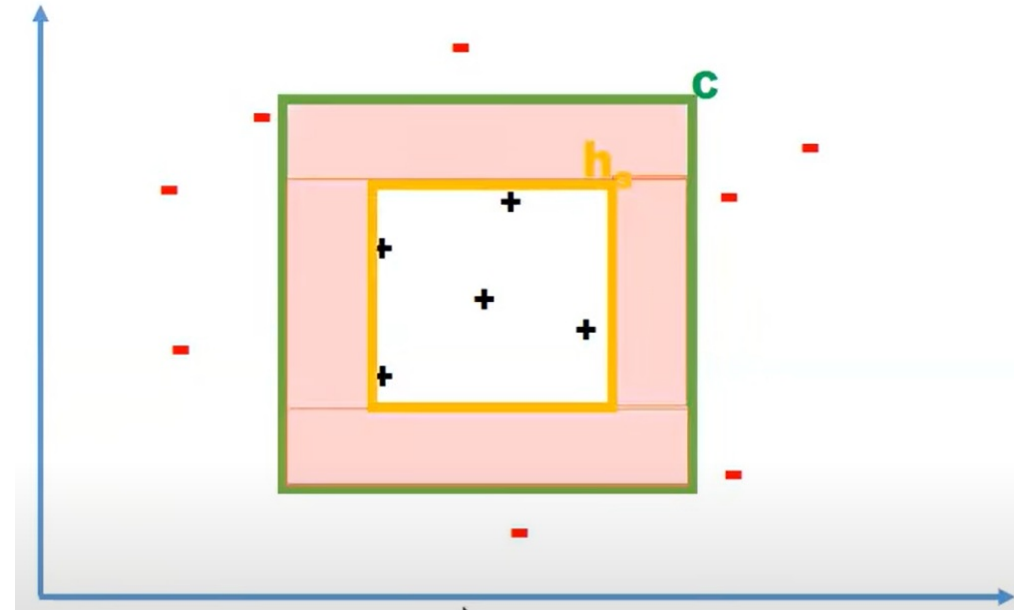


$$4(1-\varepsilon/4)^m < \delta$$

Using inequality $1-x \leq e^{-x}$

$$4(1-\varepsilon/4)^m \leq 4e^{-m\varepsilon/4} < \delta$$

$$m > \frac{4}{\varepsilon} \ln \frac{4}{\delta}$$



For example, if we want to learn a concept with 99% correctness ($\epsilon = 0.01$) with a probability of 95% ($\delta = 0.05$), the number of examples that need to be shown to our learner to learn a rectangle hypothesis is:

Version Spaces

Definition: A version space is formed by **all hypotheses(H)** that are **consistent with a given training set**, meaning they have no error on that set.

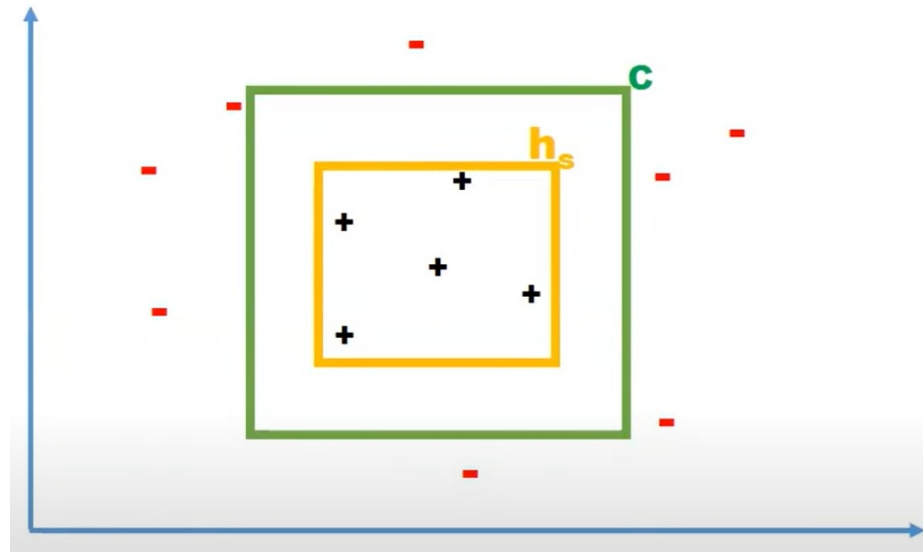
Consistency: A **hypothesis h** is considered consistent with the **training set D** if it **correctly separates all positive examples** from negative ones i.e., **$h(x) = c(x)$** .

Version Spaces

It is defined by two boundary sets:

The S-set (Most Specific Hypothesis): This represents the **tightest (smallest) possible rectangle** that encompasses all positive training examples while excluding all negative ones

The G-set (Most General Hypothesis): This represents the **largest possible rectangle** that includes all positive training examples and none of the negative ones.



Version Spaces

Any hypothesis h belonging to the hypothesis class H that lies **between the S-set and the G-set** (inclusive) is **considered a valid hypothesis, consistent** with the training set, and is thus part of the version space.

$$VS_{H,D} = \{ h \in H \mid \text{consistent}(h, D) \}$$

(Or)

$$VS_{H,D} = \{ h \in H \mid h(x)=y \text{ for all } (x, y) \in D \}$$

(Or)

$$VS_{H,D} = \{ h \in H \mid \forall (x_i, y_i) \in D, h(x_i) = y_i \}$$

Version Spaces - Algorithms

Algorithm Name	Type	Key Feature	Use Case
Candidate Elimination	Core	Maintains S and G boundaries	Efficient version space filtering
List-Then-Eliminate	Core	Brute force over hypothesis space	Simple, theoretical learning
Probabilistic Version Space	Probabilistic	Assigns probabilities to hypotheses	Learning under uncertainty
Maximum Likelihood VS (MLVS)	Probabilistic	Selects most likely consistent hypotheses	Probabilistic concept learning
Version Space Algebra (VSA)	Algebraic/Structured	Algebraic manipulation of version spaces	Program synthesis, semantic parsing
Incremental VS Learning	Online	Real-time update of version space	Online/stream learning
Active Learning + VS	Query-based	Selects examples that reduce version space	Efficient labeling
Hierarchical/ILP-based VS	Structured	Logic-based or tree-structured hypotheses	ILP, bioinformatics
Decision Tree + VS Concepts	Hybrid	Hypothesis testing via splitting	Classification
Ensemble VS Filtering	Hybrid	Filters ensemble predictions with version space	Model selection, boosting/bagging

Version Spaces - List then Eliminate Algorithm

Input:

Hypothesis space \mathbf{H} (finite and well-defined)

Training dataset $\mathbf{D} = \{ (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \}$

where each x_i is an input, and y_i is the target output

Output:

Version Space \mathbf{VS} : Set of all hypotheses consistent with all training examples

Version Spaces – List then Eliminate Algorithm

1. Initialization version space

$VS \leftarrow H$ (i.e., all hypotheses in the hypothesis space)

2. For each training example (x, y) in D :

1: For each hypothesis h in VS :

Compute prediction: $h(x_i)$

Compare with true label y_i

If $h(x_i) \neq y_i$:

Remove h from VS

This eliminates all hypotheses that are **inconsistent** with the current training example.

3. Repeat Step 2 for all examples in the dataset

4. Return final Version Space (VS)

VS contains only those hypotheses that correctly classify all training examples

Version Spaces - Candidate Elimination Algorithm

Input:

Hypothesis space H

Set of training examples $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

Output:

Final **version space** $VS_{H,D}$, bounded by sets S and G

Version Spaces - Candidate Elimination Algorithm

1. Initialization

Let $S = \{\emptyset, \emptyset, \dots, \emptyset\}$

Let $G = \{?, ?, \dots, ?\}$

Color	Shape	Texture	Fruit
Red	Round	Smooth	Yes
Green	Round	Smooth	Yes
Yellow	Oblong	Rough	No

2. For each training example (x, y) in D:

Step 1: If (x, y) is a Positive Example:

a) Remove from G any hypothesis **inconsistent** with x

b) For each hypothesis s in S:

If s does **not cover** x:

Replace s by the **minimal generalizations** of s that **cover** x

Ensure each generalization is **more specific** than some member of G

c) Remove from S any hypothesis **more general** than another hypothesis in S

Version Spaces - Candidate Elimination Algorithm

2. For each training example (x, y) in D : Cont...

Step 2: If (x, y) is a Negative Example:

a) Remove from S any hypothesis that covers x

b) For each hypothesis g in G :

If g does cover x :

Replace g by its minimal specializations that exclude x
specialization must still be more general than some member of S
from G any hypothesis less general than another in G .

Each

c) Remove

Repeat until all examples are processed.

Sample Dataset: COVID-19 Symptoms

Sample	Fever	Cough	Label
x1	Yes	Yes	Positive
x2	Yes	No	Positive
x3	No	Yes	Negative
x4	No	No	Negative

**Example Dataset: Weather-based
Play Decision**

Sample	Weather	Temperature	Play
x1	Sunny	Hot	Yes
x2	Sunny	Mild	Yes
x3	Rainy	Hot	No
x4	Rainy	Mild	No

Training Dataset Recap

Size	Color	Shape	Class
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes


Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



Role of Machine Learning in Artificial Intelligence

It is considered a cornerstone for creating intelligent systems that can learn and adapt to dynamic environments.

a) Enabling Adaptability and Intelligence

- ML involves computational methods that use "**experience**" (past data) to improve performance or make accurate predictions.
 - Allows the system to **learn and adapt without requiring explicit programming** for every conceivable situation .
- 

Criteria	Use Machine Learning	Do Not Use Machine Learning
Complexity of the problem	Problem is too complex for rules or equations	Simple logic or rules can solve the problem
Availability of data	Sufficient historical or training data is available	No or very limited data is available
Pattern recognition	Need to discover patterns or relationships automatically	Patterns are already known and can be hardcoded
Adaptability requirement	System needs to adapt/improve over time	No need for adaptation or learning
Generalization needed	Must perform well on new, unseen data	Only works with a fixed set of known inputs


Criteria	Use Machine Learning	Do Not Use Machine Learning
Rule design feasibility	Rules are hard or impossible to define manually	Rules can be easily written by humans
Accuracy over time	Acceptable to start with moderate accuracy and improve	Immediate, perfect accuracy is required
Resources (time, budget, expertise)	Resources are available for model building, training, and maintenance	Limited time, budget, or expertise for ML
System behavior	Data or environment is dynamic and constantly changing	Static data or fixed processes



Role of Machine Learning in Artificial Intelligence

It is considered a cornerstone for creating intelligent systems that can learn and adapt to dynamic environments.


a) Enabling Adaptability and Intelligence

- ML involves computational methods that use "**experience**" (past data) to improve performance or make accurate predictions.
 - Allows the system to **learn and adapt without requiring explicit programming** for every conceivable situation .
- 



Role of Machine Learning in Artificial Intelligence

b) Solving Complex, Data-Driven Problems

- ML offers solutions to **problems that are inherently difficult or impossible** to program using traditional methods due to the vastness and variability of data.
 - ML algorithms combine fundamental concepts with ideas from **statistics, probability, and optimisation**.
 - Designed to **efficiently detect patterns** in large datasets and then utilize these patterns to **make future predictions** or inform decisions.
- 



Role of Machine Learning in Artificial Intelligence

c) Key Machine Learning Paradigms Contributing to AI

- Supervised Learning.
- Unsupervised Learning
- Reinforcement Learning (RL)

d) Advancing AI Capabilities

- Machine learning success has reshaped AI thinking
 - ML bridges multiple disciplines
- 