MODULE:1

Introduction to Machine Learning

Machine Learning and its Applications – Learning Problems –

Designing a Learning System – Perspectives and Issues in

Machine Learning - Version Spaces – Finite and Infinite Hypothesis

Spaces – PAC Learning

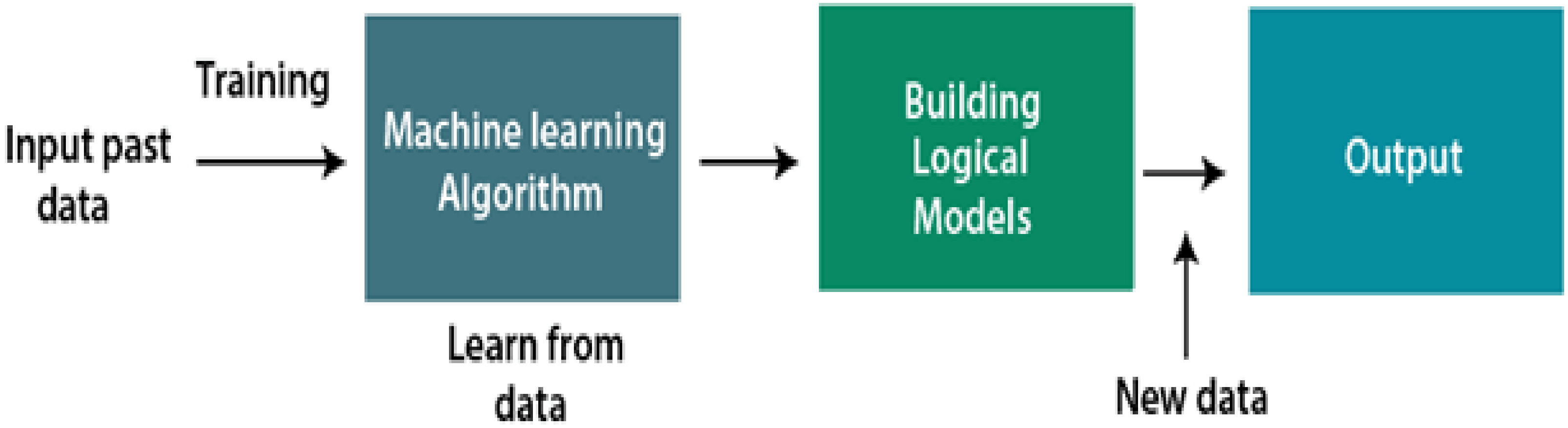
WHAT IS MACHINE LEARNING?

Machine learning is a branch of artificial intelligence that enables computers **to learn from data, identify patterns, and make decisions or predictions without explicit programming**. It involves algorithms that improve their performance on a specific task through experience.

WHY “LEARN” ?

- There is no need to “learn” to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

HOW DOES MACHINE LEARNING WORK



FEATURES OF MACHINE LEARNING

- **Learning from Data:** Machine learning systems learn from data, adjusting their internal parameters to improve performance on a specific task as more data becomes available.
- **Adaptability:** Machine learning algorithms can adapt and evolve over time as they encounter new data, allowing them to handle a variety of situations without explicit programming changes.
- **Generalization:** Machine learning models aim to generalize patterns learned from training data to make accurate predictions or decisions on new, unseen data.
- **Pattern Recognition:** ML algorithms excel at identifying complex patterns and relationships within large datasets, which may be challenging for humans to discern.
- **Automation:** Once trained, machine learning models can automate decision-making processes, reducing the need for explicit programming for specific tasks.

FEATURES OF MACHINE LEARNING CONT…

- **Prediction and Classification:** ML models can make predictions about future outcomes or classify data into predefined categories based on patterns learned during training.
- **Feedback Loop:** Many machine learning systems operate in a feedback loop, continuously learning and adapting based on the feedback received from their predictions or decisions.
- **Scalability:** Machine learning techniques can handle large and diverse datasets, making them suitable for applications ranging from simple tasks to complex problems.
- **Unsupervised Learning:** Some machine learning approaches, such as unsupervised learning, allow systems to discover patterns and relationships in data without labeled examples.
- **Personalization:** Machine learning enables systems to personalize experiences by tailoring responses or recommendations based on individual user behavior and preferences.

APPLICATIONS OF MACHINE LEARNING

- **Image and Speech Recognition:**
  - Identifying objects, faces, and patterns in images.
  - Transcribing and understanding spoken language.
- **Natural Language Processing (NLP):**
  - Language translation services.
  - Sentiment analysis in social media and customer reviews.
  - Chatbots and virtual assistants.
- **Healthcare:**
  - D isease diagnosis and prediction.
  - Personalized medicine and treatment recommendations.
  - Drug discovery and development.

- **Finance:**
  - Credit scoring and risk assessment.
  - Fraud detection and prevention.
  - Algorithmic trading and stock market predictions.
- **E-commerce and Recommendation Systems:**
  - Product recommendations based on user behavior.
  - Personalized marketing strategies.
- **Autonomous Vehicles:**
  - Object detection and recognition for self-driving cars.
  - Path planning and navigation.

- **Cyber-security:**
  - Anomaly detection and threat analysis.
  - Identifying and preventing security breaches.
- **Manufacturing and Supply Chain:**
  - Predictive maintenance for machinery.
  - Demand forecasting and inventory optimization.
- **Energy Management:**
  - Predictive maintenance for equipment in the energy sector.
  - Energy consumption optimization.

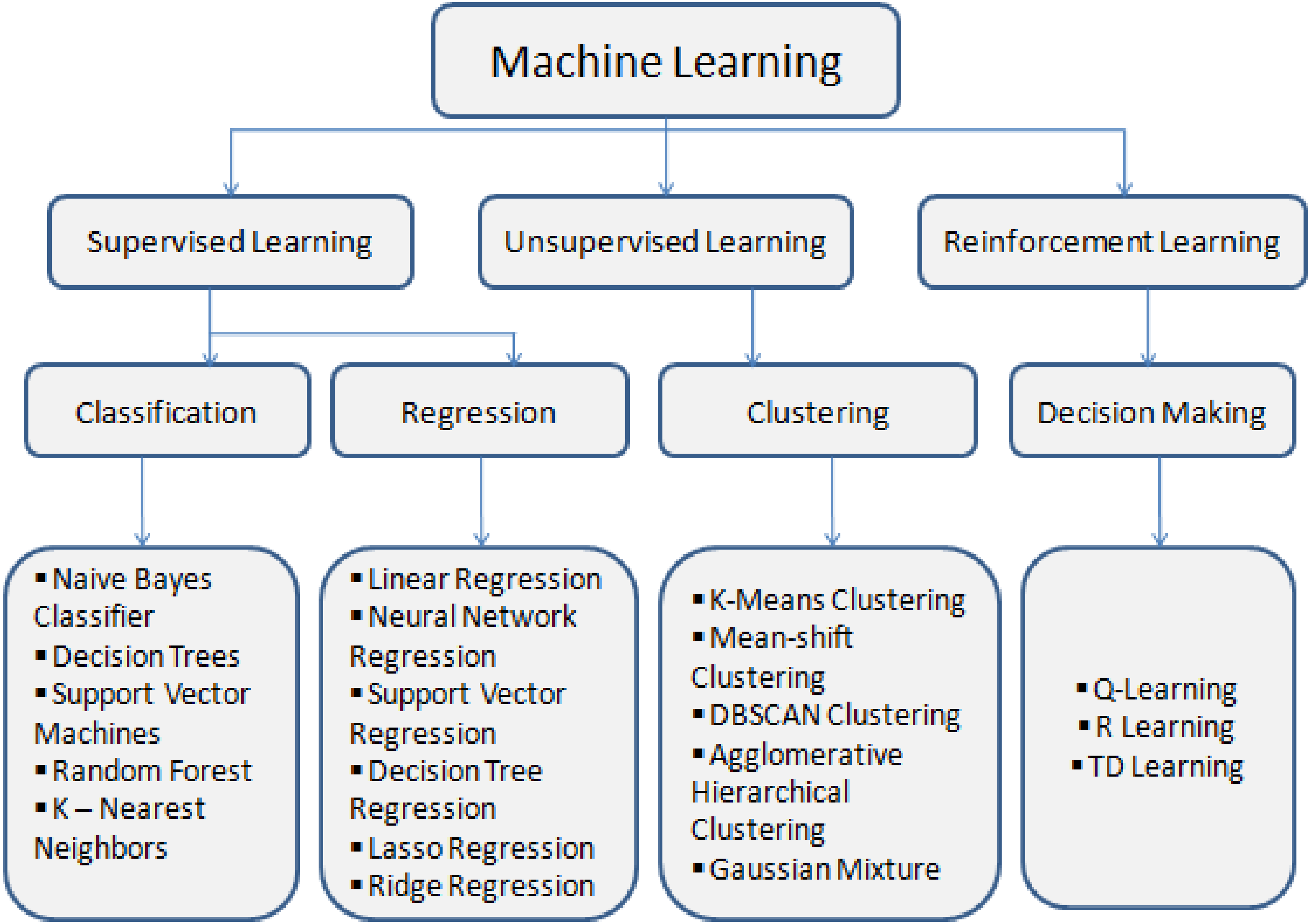
- **Education:**
    - Adaptive learning platforms.
    - Student performance prediction and intervention.
- **Human Resources:**
    - Resume screening and candidate matching.
    - Employee turnover prediction.
- **Environmental Monitoring:**
    - Climate modeling and prediction.
    - Air and water quality monitoring.

- **Gaming:**
  - Creating intelligent non-player characters (NPCs).
  - Personalized gaming experiences based on player behavior.
- **Marketing and Customer Segmentation:**
  - Customer segmentation for targeted marketing.
  - Predicting customer churn.
- **Social Media:**
  - Content recommendation.
  - Social network analysis

LEARNING PROBLEMS

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

- **Supervised Learning** the algorithm is trained on a labeled dataset, where the input data is paired with corresponding output labels.
  - **Example:** Training a model to recognize spam emails using a dataset where each email is labeled as either spam or not spam.
- **Unsupervised Learning** involves algorithms learning from unlabeled data to discover patterns, relationships, or structures within the data.
  - **Example:** Clustering similar customer purchase behavior without having predefined categories or labels.
- **Reinforcement Learning** is about training agents to make sequential decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties.
  - **Example:** Training a computer program to play and improve at a game by receiving rewards (points) or penalties based on its actions in the game environment.



Machine Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning

Classification

Regression

Clustering

Decision Making

- Naive Bayes Classifier
- Decision Trees
- Support Vector Machines
- Random Forest
- K – Nearest Neighbors

- Linear Regression
- Neural Network Regression
- Support Vector Regression
- Decision Tree Regression
- Lasso Regression
- Ridge Regression

- K-Means Clustering
- Mean-shift Clustering
- DBSCAN Clustering
- Agglomerative Hierarchical Clustering
- Gaussian Mixture

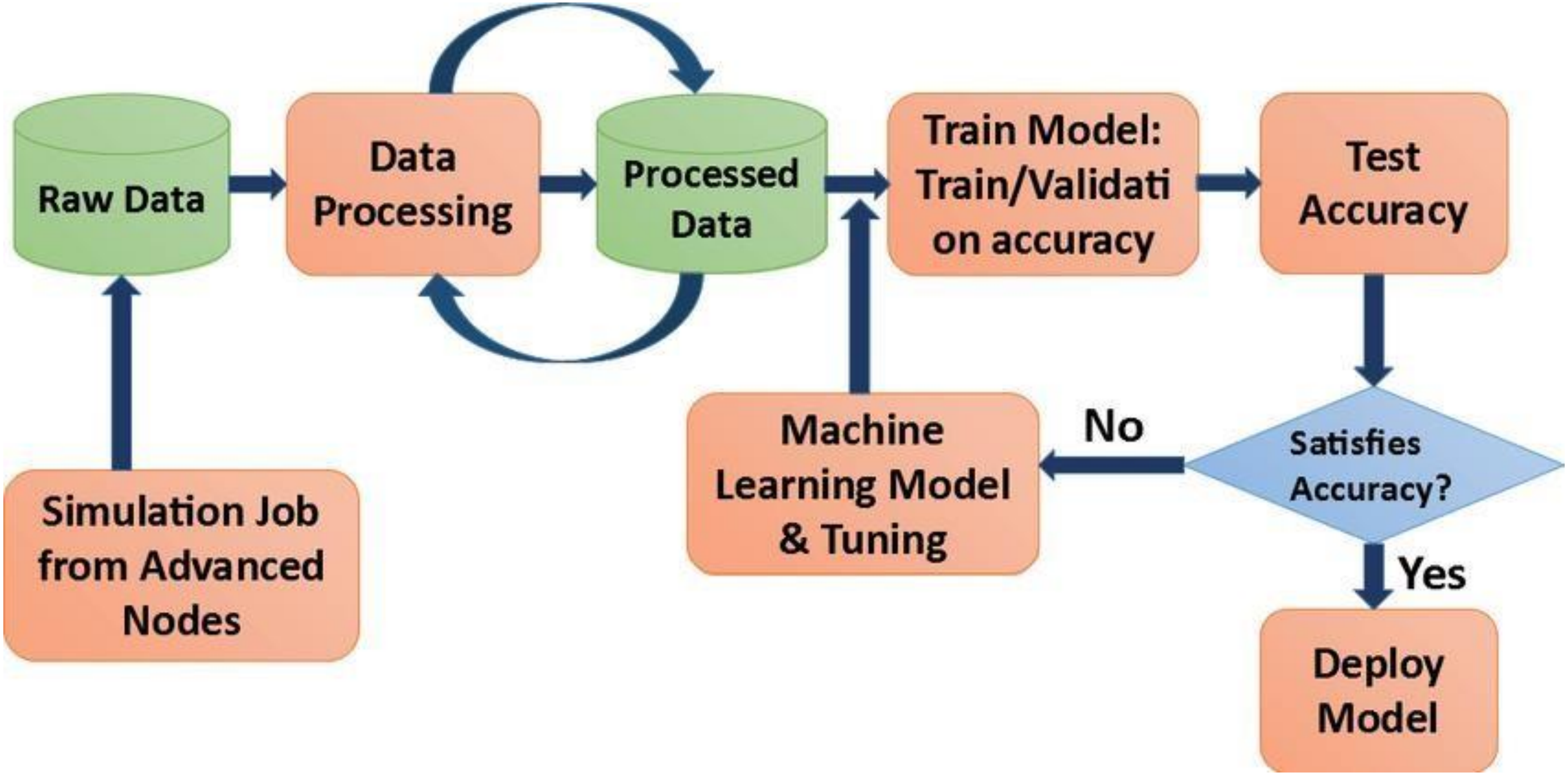
- Q-Learning
- R Learning
- TD Learning

DESIGNING A LEARNING SYSTEM IN MACHINE LEARNING

1. Identify the type of data that will be used
2. Determine the desired outcome of the learning system
3. Resources available for the learning system must be considered
4. Select a machine-learning algorithm and begin the training process
5. After the learning system is trained, it is important to fine-tune the model by adjusting the parameters and hyperparameters

Qualities that you need to keep in mind

- **Reliability**
- **Scalability**
- **Maintainability**
- **Adaptability**


Raw Data
Data Processing
Processed Data
Train Model: Train/Validati on accuracy
Test Accuracy
Simulation Job from Advanced Nodes
Machine Learning Model & Tuning
No
Satisfies Accuracy?
Yes
Deploy Model

PERSPECTIVES IN ML

- **Positive Impact on Industries:**
  - Machine learning has brought about transformative changes across industries, improving efficiency, decision-making, and user experiences.
- **Innovation and Research:**
  - Machine learning drives innovation and fosters research in diverse fields, leading to new applications and discoveries.
- **Automation and Optimization:**
  - ML enables automation of tasks, leading to increased productivity and optimization of processes in various sectors.
- **Personalization and User Experience:**
  - ML powers personalized recommendations, enhancing user experiences in services like streaming platforms, e-commerce, and social media.

ISSUES IN ML

Bias and Fairness:

- Machine learning models can inherit biases present in training data, leading to unfair or discriminatory outcomes, especially in areas like hiring and law enforcement.

Explainability and Interpretability:

- Many complex machine learning models lack transparency, making it challenging to understand how they reach specific decisions, which can be a concern in critical applications like healthcare and finance.

Data Privacy:

- Machine learning often relies on large datasets, raising concerns about privacy and the responsible use of personal information.

Security:

- ML models can be vulnerable to adversarial attacks, where malicious actors manipulate input data to deceive the model and cause incorrect predictions.

Ethical Considerations:

- Ethical concerns arise in areas such as the use of AI in autonomous weapons, surveillance technologies, and other applications with potential societal impacts.

Regulatory Challenges:

- Developing and implementing regulations for machine learning poses challenges due to the rapid pace of technological advancement and the global nature of data.

Data Quality and Quantity:

- The performance of machine learning models is highly dependent on the quality and quantity of the training data, and obtaining diverse and representative datasets can be a challenge.

Overfitting and Generalization:

- ML models may become too specialized and perform poorly on new, unseen data (overfitting) or fail to generalize well to diverse scenarios.

Environmental Impact:

- The computational demands of training large models contribute to concerns about the environmental impact, particularly in terms of energy consumption.

Lack of Domain Expertise:

- Developing effective machine learning solutions requires expertise not only in machine learning but also in the specific domain of application, and the lack of such expertise can hinder successful implementation.

VERSION SPACE

- Represents the set of all possible **hypotheses** that are **consistent** with the training data.
- Helps narrow down the possible solutions based on the observed examples, guiding the learning algorithm towards a more accurate model

HYPOTHESIS

- Hypothesis is a function or model that the algorithm is trying to learn from the provided data.
- This function represents a potential solution to the learning task, mapping input features to output predictions.
- For example, in linear regression, a hypothesis might be a linear equation representing the relationship between input features and the target variable. In a more complex model like a neural network, the hypothesis involves the specific architecture and weights that define the network's behavior.
- **The process of machine learning often involves refining and updating hypotheses based on the observed data to improve the model's accuracy and ability to generalize to new, unseen data. The ultimate goal is to find a hypothesis that accurately represents the underlying patterns in the data.**

HYPOTHESIS SPACE

- Refers to the set of all possible hypotheses or models that a learning algorithm can potentially output.
- It represents the range of functions or relationships between input and output that the algorithm considers during the learning process.
- **The goal is to find the best hypothesis within this space that accurately captures the underlying patterns in the data.**

FINITE & INFINITE HYPOTHESIS SPACE

Finite Hypothesis Space:

- This type has a limited, countable number of possible hypotheses.
- The learning algorithm considers a finite set of models during the training process.
- Common in situations where there are constraints on the complexity of the models.

Infinite Hypothesis Space:

- Has an uncountable number of possible hypotheses.
- The learning algorithm can potentially consider an unlimited variety of models.
- Often associated with more complex and flexible models, such as neural networks with many parameters.

- The choice between finite and infinite hypothesis spaces depends on the problem at hand, computational considerations, and the nature of the data.
- Finite spaces might be preferred for simplicity and efficiency, while infinite spaces can capture more complex relationships but may require careful regularization to avoid overfitting.

CONSISTENT HYPOTHESIS

- A hypothesis is considered consistent with the training data if it correctly predicts the outcomes for all examples in the dataset.
- Essentially, the hypothesis aligns with the provided training examples without making any errors on them.

FIND-S ALGORITHM

- The Find-S algorithm is a concept from machine learning, specifically within the context of supervised learning and concept learning.
- It's a simple algorithm used for learning a hypothesis from a set of training examples.

- **Initialization:** Start with the most specific hypothesis. In the context of binary attributes, this means initializing the hypothesis to be the most specific hypothesis possible, which is a hypothesis that says nothing is true.
- **Iterative Refinement:** For each positive training example, update the hypothesis to include the positive example. The idea is to generalize the hypothesis to cover the positive examples.

  For each negative training example, there is no change to the hypothesis because negative examples are used to specify what the hypothesis should not cover.
- **Final Hypothesis:** The final hypothesis is the most specific hypothesis consistent with the training examples. This means that the hypothesis includes all the positive examples and excludes as much as possible from the negative examples.

EXAMPLE -1

Sky	Air Temp	Humidity	Wind	Weather	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Warm	Same	YES
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Important Representation :

? indicates that any value is acceptable for the attribute.

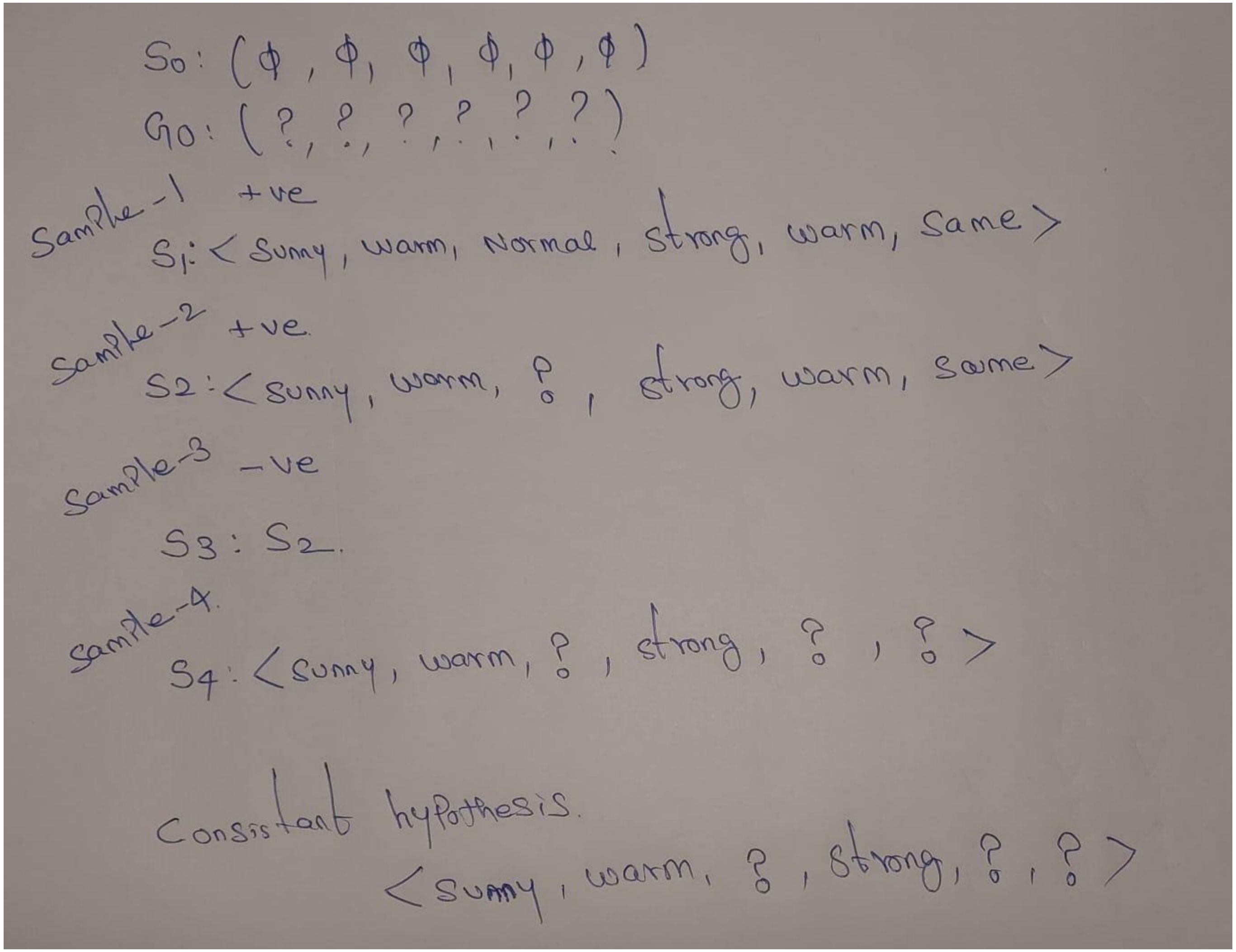
specify a single required value (e.g., Cold) for the attribute.

φindicates that no value is acceptable.

The most **general hypothesis** is represented by: **{?, ?, ?, ?, ?, ?}**

The most **specific hypothesis** is represented by: **{φ, φ, φ, φ, φ, φ}**

SOLUTION



$S_0 : (\phi, \phi, \phi, \phi, \phi, \phi)$

$G_0 : (?, ?, ?, ?, ?, ?)$

Sample-1 +ve

S_1: < Sunny, warm, Normal, strong, warm, Same >

Sample-2 +ve

S_2 :< sunny, warm, ?, strong, warm, same >

Sample-3 –ve

$S_3 : S_2$.

sample-4.

S_4: < sunny, warm, ?, strong, ?, ? >

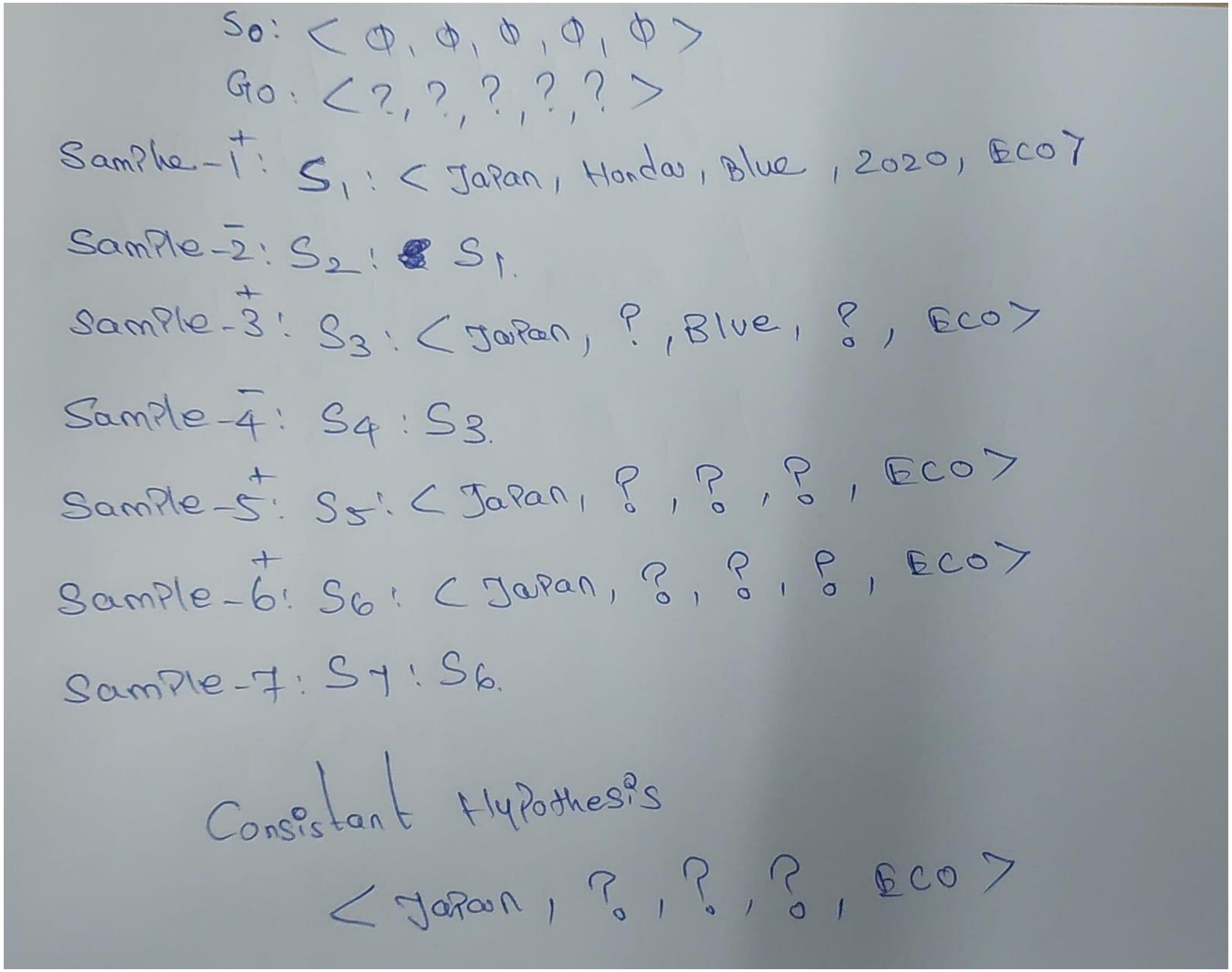
Consistant hypothesis.

< sunny, warm, ?, strong, ?, ? >

EXAMPLE -2

Origin	Manufacturer	Color	Year	Type	Buy
Japan	Honda	Blue	2020	Eco	Yes
Japan	Toyota	Green	2022	Sport	No
Japan	Toyota	Blue	2019	Eco	Yes
USA	Audi	Red	2018	Eco	No
Japan	Honda	White	2023	Eco	Yes
Japan	Toyota	Green	2016	Eco	Yes
Japan	Honda	Red	2017	Eco	No

SOLUTION



S_0: < ϕ, ϕ, ϕ, ϕ, ϕ >

G_0: < ?, ?, ?, ?, ? >

Sample-$\overset{+}{1}$: S_1: < Japan, Honda, Blue, 2020, ECO >

Sample-$\overset{-}{2}$: S_2: S_1.

Sample-$\overset{+}{3}$: S_3: < Japan, ?, Blue, ?, ECO >

Sample-$\overset{-}{4}$: S_4 : S_3.

Sample-$\overset{+}{5}$: S_5: < Japan, ?, ?, ?, ECO >

Sample-$\overset{+}{6}$: S_6: < Japan, ?, ?, ?, ECO >

Sample-7: S_7: S_6.

Consistant Hypothesis

< Japan, ?, ?, ?, ECO >

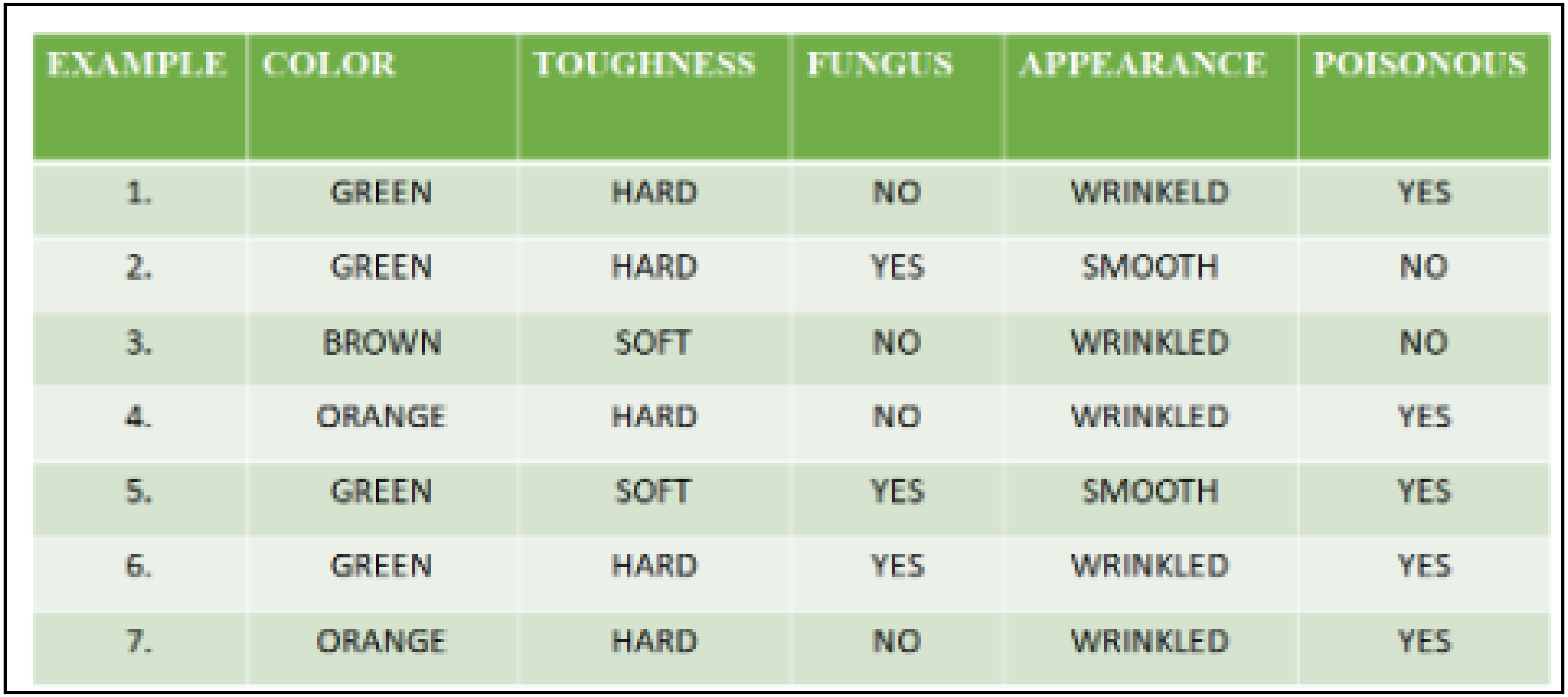
TASK-1

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

TASK-2

Outlook	Temp	Humidity	Windy	Play
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Rainy	Mild	High	True	No

Task 3



EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES

CANDIDATE ELIMINATION ALGORITHM

- The Candidate Elimination algorithm is a concept learning algorithm used in machine learning.
- It is designed to incrementally build a general, consistent hypothesis that fits the observed training data.
- The algorithm maintains both a specific and a general hypothesis throughout the learning process.
- The Candidate Elimination algorithm is particularly useful in scenarios where the target concept can be expressed in terms of a conjunction of attribute values.
- It's a form of incremental learning that refines hypotheses as more training data becomes available, aiming to converge to a final, accurate hypothesis consistent with the observed data.

OVERVIEW OF HOW THE CANDIDATE ELIMINATION ALGORITHM WORKS

- **Initialize:**

Start with the most specific hypothesis (hypothesis that includes all possible values for each attribute) and the most general hypothesis (hypothesis that includes no specific values for any attribute).

- **Iterate Through Training Examples:**

For each training example, update the specific and general hypotheses based on whether the example is classified as positive or negative.

- **Refine Hypotheses:**

If an example is classified as positive, generalize the specific hypothesis by incorporating the observed positive instance's attribute values.

Conversely, if an example is classified as negative, specialize the general hypothesis by excluding the observed negative instance's attribute values.

- **Consistency Check:**

Ensure that the specific hypothesis remains consistent with positive instances, and the general hypothesis remains consistent with negative instances.

- **Repeat:**

Iterate through the training examples, updating and refining the hypotheses until a consistent and minimal version space is obtained.

Terms Used:

Concept learning: Concept learning is basically the learning task of the machine (Learn by Train data)

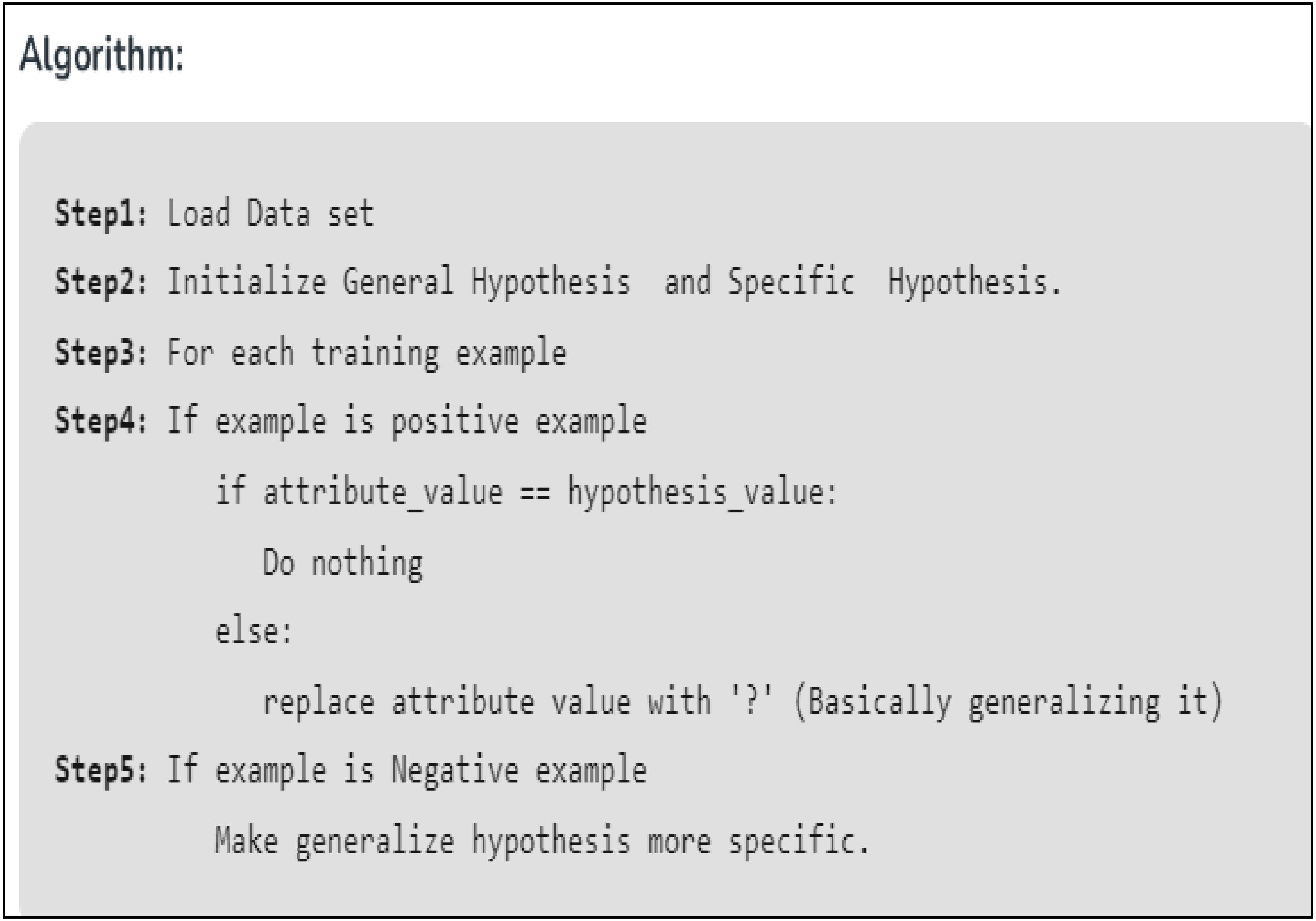
General Hypothesis: Not Specifying features to learn the machine.

G = {'?', '?','?','?'…}: Number of attributes

Specific Hypothesis: Specifying features to learn machine (Specific feature)

S= {'pi','pi','pi'…}: The number of pi depends on a number of attributes.

Version Space: It is an intermediate of general hypothesis and Specific hypothesis. It not only just writes one hypothesis but a set of all possible hypotheses based on training data-set.



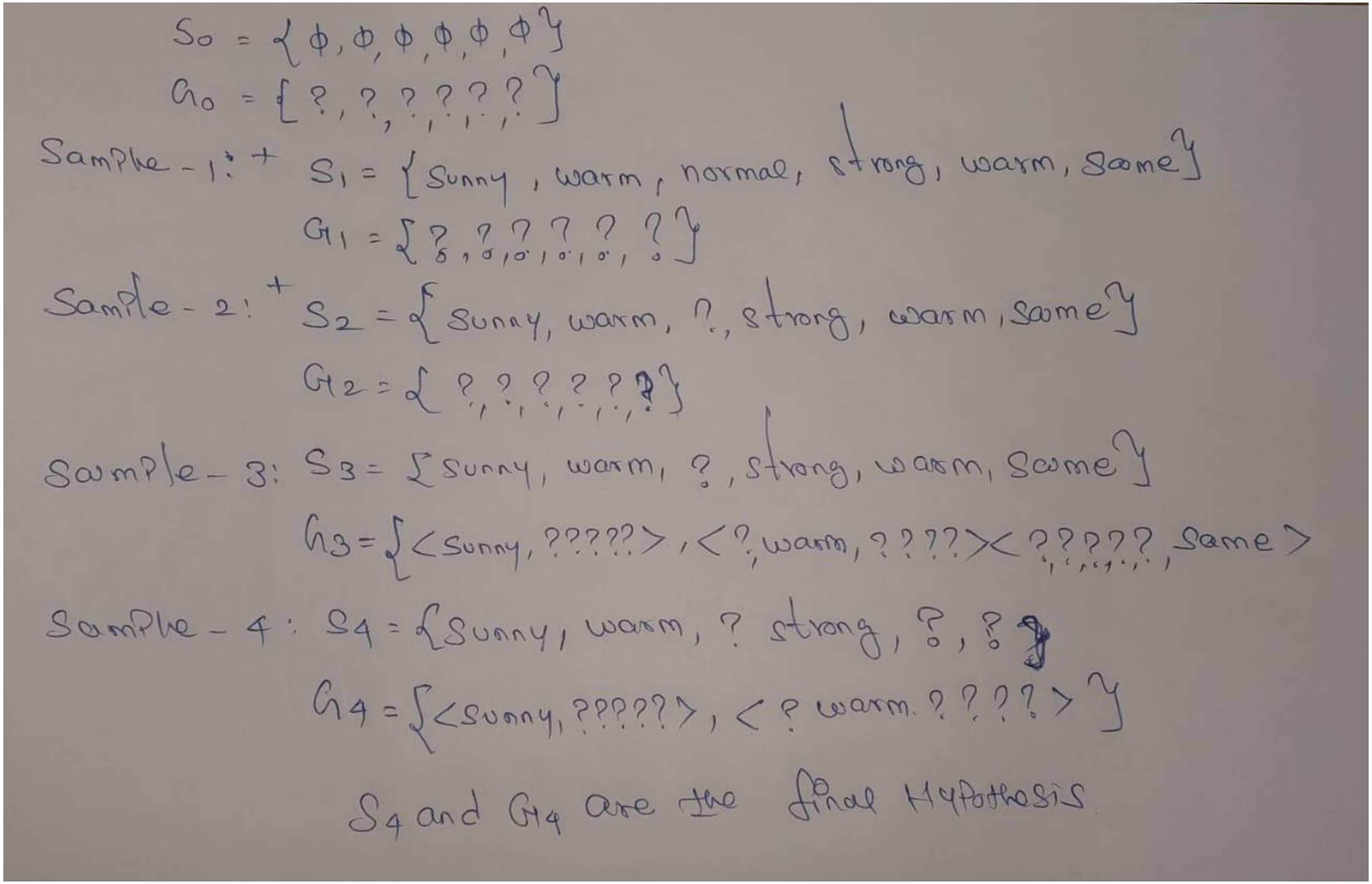
Algorithm:

```
Step1: Load Data set

Step2: Initialize General Hypothesis  and Specific  Hypothesis.

Step3: For each training example

Step4: If example is positive example

          if attribute_value == hypothesis_value:

             Do nothing

          else:

             replace attribute value with '?' (Basically generalizing it)

Step5: If example is Negative example

          Make generalize hypothesis more specific.
```

EXAMPLE -1

Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
Sunny	Warm	Normal	Strong	Warm	Same	YES
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

SOLUTION



S_0 = {ϕ, ϕ, ϕ, ϕ, ϕ, ϕ}

G_0 = [?, ?, ?, ?, ?, ?]

Sample - 1: + S_1 = {Sunny, warm, normal, strong, warm, Same}

G_1 = {?, ?, ?, ?, ?, ?}

Sample - 2: + S_2 = {Sunny, warm, ?, strong, warm, Same}

G_2 = {?, ?, ?, ?, ?, ?}

Sample - 3: S_3 = {Sunny, warm, ?, strong, warm, Same}

G_3 = {<Sunny, ?????>, <?, warm, ????> <?, ?, ?, ?, ?, Same>

Sample - 4: S_4 = {Sunny, warm, ? strong, ?, ?}

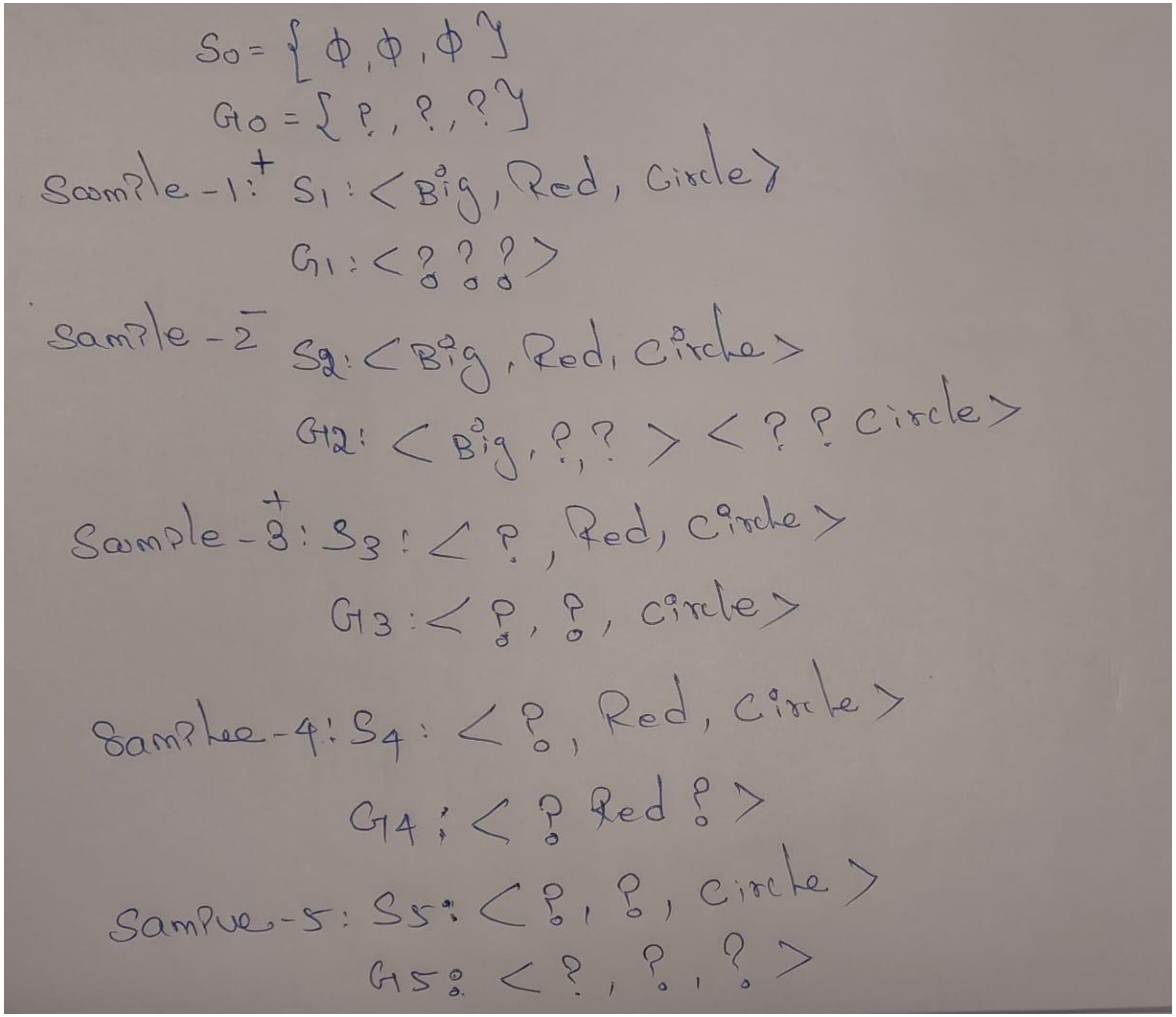
G_4 = {<Sunny, ?????>, <? warm. ????>}

S_4 and G_4 are the final Hypothesis

EXAMPLE -2

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

SOLUTION



S_0 = {Φ, Φ, Φ}

G_0 = {?, ?, ?}

Sample-1:$^+$ S_1: <Big, Red, Circle>

G_1: <? ? ?>

Sample-2$^-$ S_2: <Big, Red, circle>

G_2: <Big, ?, ? > < ? ? circle>

Sample-3$^+$: S_3: < ?, Red, circle >

G_3: < ?, ?, circle>

Samplee-4: S_4: < ?, Red, circle >

G_4: < ? Red ? >

Sampue-5: S_5: < ?, ?, circle >

G_5: < ?, ?, ? >

TASK - 1

Origin	Manufacturer	Color	Year	Type	Buy
Japan	Honda	Blue	2020	Eco	Yes
Japan	Toyota	Blue	2019	Eco	Yes
USA	Audi	Red	2018	Eco	No
Japan	Honda	White	2023	Eco	Yes
Japan	Toyota	Green	2016	Eco	Yes

TASK-2

Outlook	Temp	Humidity	Windy	Play
Sunny	Mild	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Mild	Normal	False	Yes
Rainy	Cool	Normal	True	No
Sunny	Mild	Normal	False	Yes
Sunny	Cool	High	True	No
Sunny	Mild	Normal	False	Yes

COMPUTATIONAL LEARNING THEORY

- Are there general laws that govern learning?
  - ***Sample Complexity:*** How many training examples are needed to learn a successful hypothesis?
  - ***Computational Complexity:*** How much computational effort is needed to learn a successful hypothesis?
  - ***Mistake Bound:*** How many training examples will the learner misclassify before converging to a successful hypothesis?

X is the set of all possible instances

C is the set of all possible concepts c where $c : X \rightarrow \{0,1\}$

H is the set of hypotheses considered by a learner, $H \subseteq C$

L is the learner

D is a probability distribution over X that generates observed instances

- The ***true error*** of hypothesis *h*, with respect to the target concept *c* and observation distribution *D* is the probability that *h* will misclassify an instance drawn according to *D*

$$error_D \equiv \underset{x \in D}{P}[c(x) \neq h(x)]$$

- In a perfect world, we'd like the true error to be 0

THE WORLD ISN'T PERFECT

- We typically can't provide every instance for training.
- Since we can't , there is always a chance the examples provided the learner will be misleading
  - "No Free Lunch" theorem (no single machine learning algorithm is universally the best-performing algorithm for all problems)
- So we'll go for a weaker thing:

PROBABLY APPROXIMATELY CORRECT learning

A concept class **C** is “PAC learnable” by a hypothesis class **H** iff there exists a learning algorithm **L** such that..

…. given any target concept **c** in **C** ,

any target distribution **D** over the possible examples **X** ,

and any pair of real numbers 0< ε, δ <1

… that **L** takes as input a training set of **m** examples drawn according to **D**, where the size of **m** is bounded above by a polynomial in 1/ε and 1/δ

… and outputs an hypothesis **h** in **H** about which we can say, with confidence (probability over all possible choices of the training set) greater than 1 − δ

…. that the error of the hypothesis is less than ε.

$$error_D \equiv \underset{x \in D}{P}[c(x) \neq h(x)] \leq \varepsilon$$

- A hypothesis is ***consistent*** with the training data if it returns the correct classification for every example presented it.
- A ***consistent learner*** returns only hypotheses that are consistent with the training data.
- Given a consistent learner, the number of examples sufficient to assure that any hypothesis will be probably (with probability ***(1-*** δ***)***) approximately (within error ε) correct is...

$$m \geq \frac{1}{\varepsilon}\left(\ln|H| + \ln(1/\delta)\right)$$

Theorem:
If the hypothesis space H is finite, and D is a sequence of $m \geq 1$ independent random examples of some target concept c, then for any $0 \leq \epsilon \leq 1$, the probability that $VS_{H,D}$ contains a hypothesis with error greater than ϵ is less than

$$|H|e^{-\epsilon m}$$

Proof sketch:
Prob(1 hyp. w/ error $> \epsilon$ consistent w/ 1 ex.) $< 1 - \epsilon \leq e^{-\epsilon}$
Prob(1 hyp. w/ error $> \epsilon$ consistent with m exs.) $< e^{-\epsilon m}$
Prob(1 of $|H|$ hyps. consistent with m exs.) $< |H|e^{-\epsilon m}$

Interesting! This bounds the probability that any consistent learner will output a hypothesis h with $error(h) \geq \epsilon$

If we want this probability to be at most δ

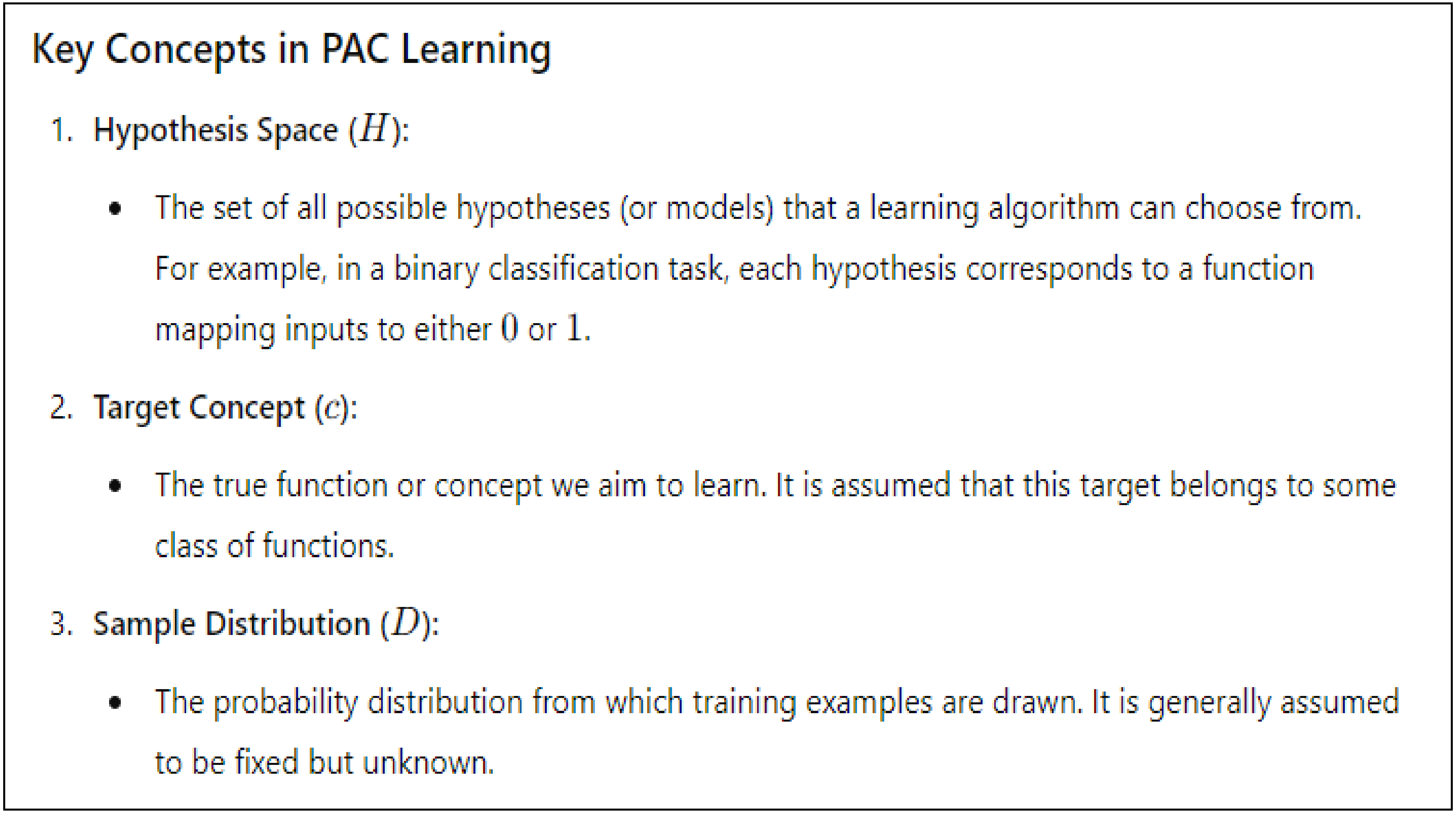
$$|H|e^{-\epsilon m} \leq \delta$$

then

$$m \geq \frac{1}{\epsilon}(\ln|H| + \ln(1/\delta))$$

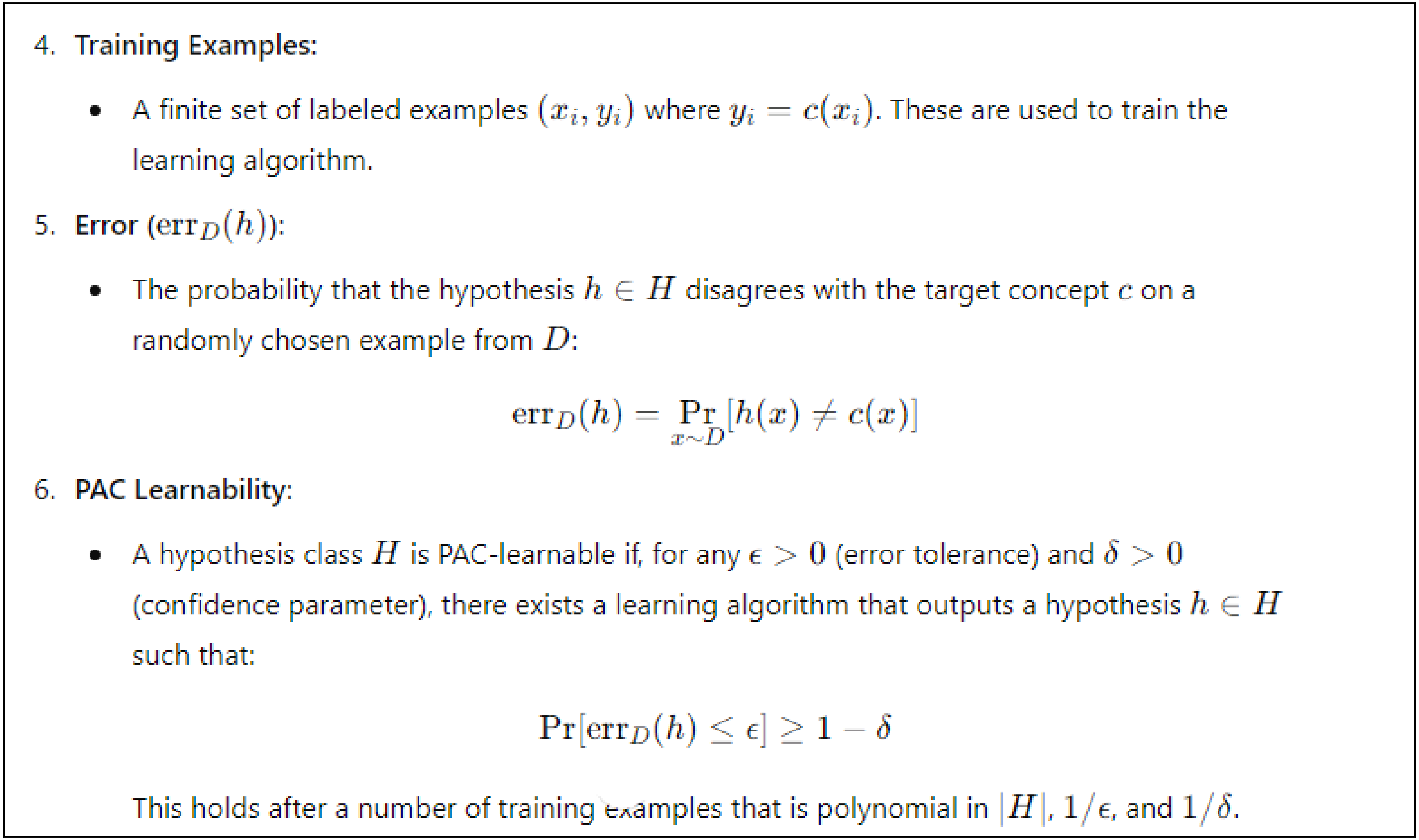
PAC Learning

Probably Approximately Correct (PAC) learning is a fundamental framework in computational learning theory introduced by Leslie Valiant in 1984. It provides a rigorous mathematical framework to study how well a learning algorithm can generalize from a limited set of training examples to unseen examples.



Key Concepts in PAC Learning

1. **Hypothesis Space (H):**
   - The set of all possible hypotheses (or models) that a learning algorithm can choose from. For example, in a binary classification task, each hypothesis corresponds to a function mapping inputs to either 0 or 1.
2. **Target Concept (c):**
   - The true function or concept we aim to learn. It is assumed that this target belongs to some class of functions.
3. **Sample Distribution (D):**
   - The probability distribution from which training examples are drawn. It is generally assumed to be fixed but unknown.



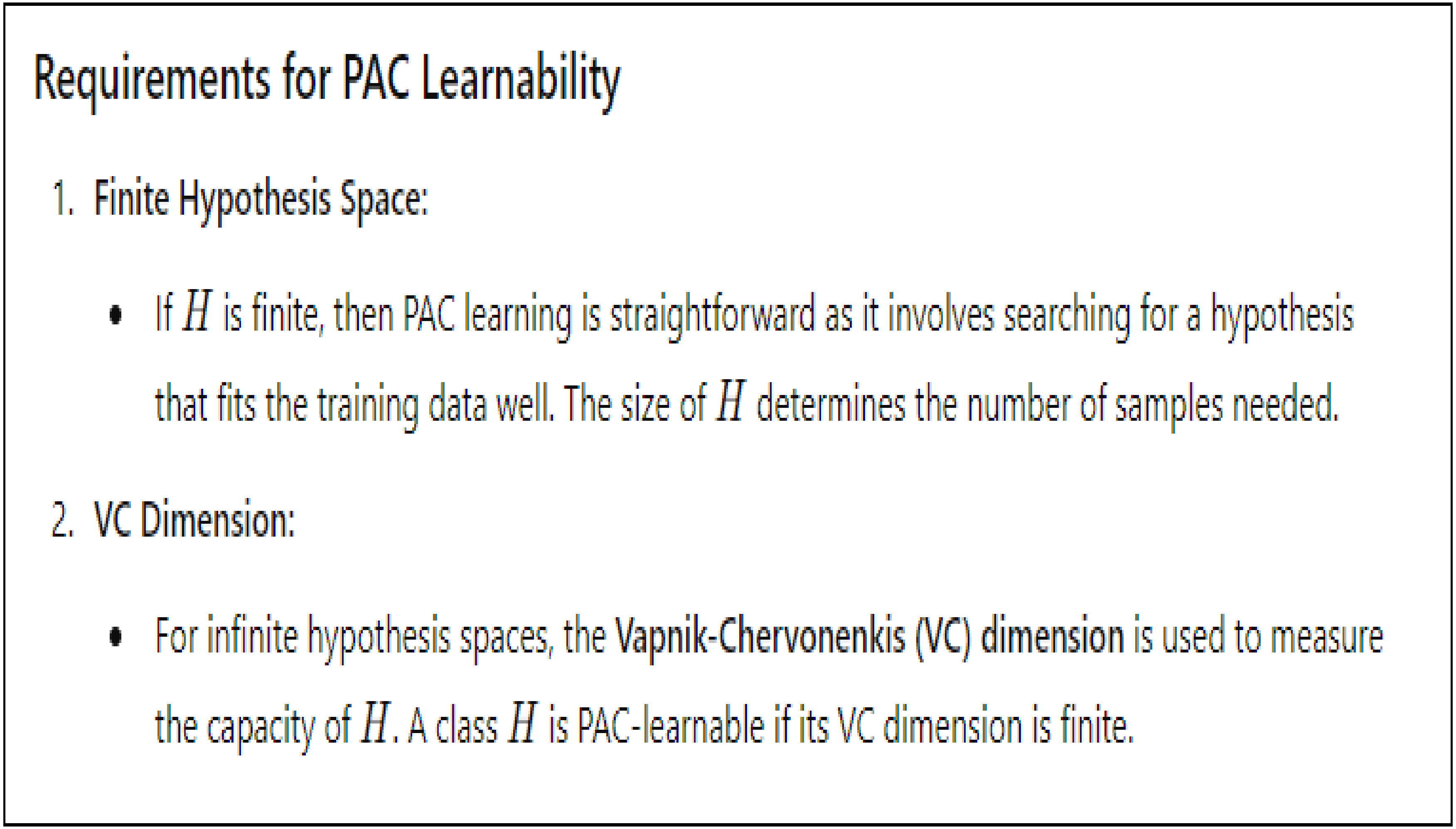
4. **Training Examples:**
   - A finite set of labeled examples (x_i, y_i) where $y_i = c(x_i)$. These are used to train the learning algorithm.
5. **Error** ($\mathrm{err}_D(h)$):
   - The probability that the hypothesis $h \in H$ disagrees with the target concept c on a randomly chosen example from D:

$$\mathrm{err}_D(h) = \Pr_{x \sim D}[h(x) \neq c(x)]$$

6. **PAC Learnability:**
   - A hypothesis class H is PAC-learnable if, for any $\epsilon > 0$ (error tolerance) and $\delta > 0$ (confidence parameter), there exists a learning algorithm that outputs a hypothesis $h \in H$ such that:

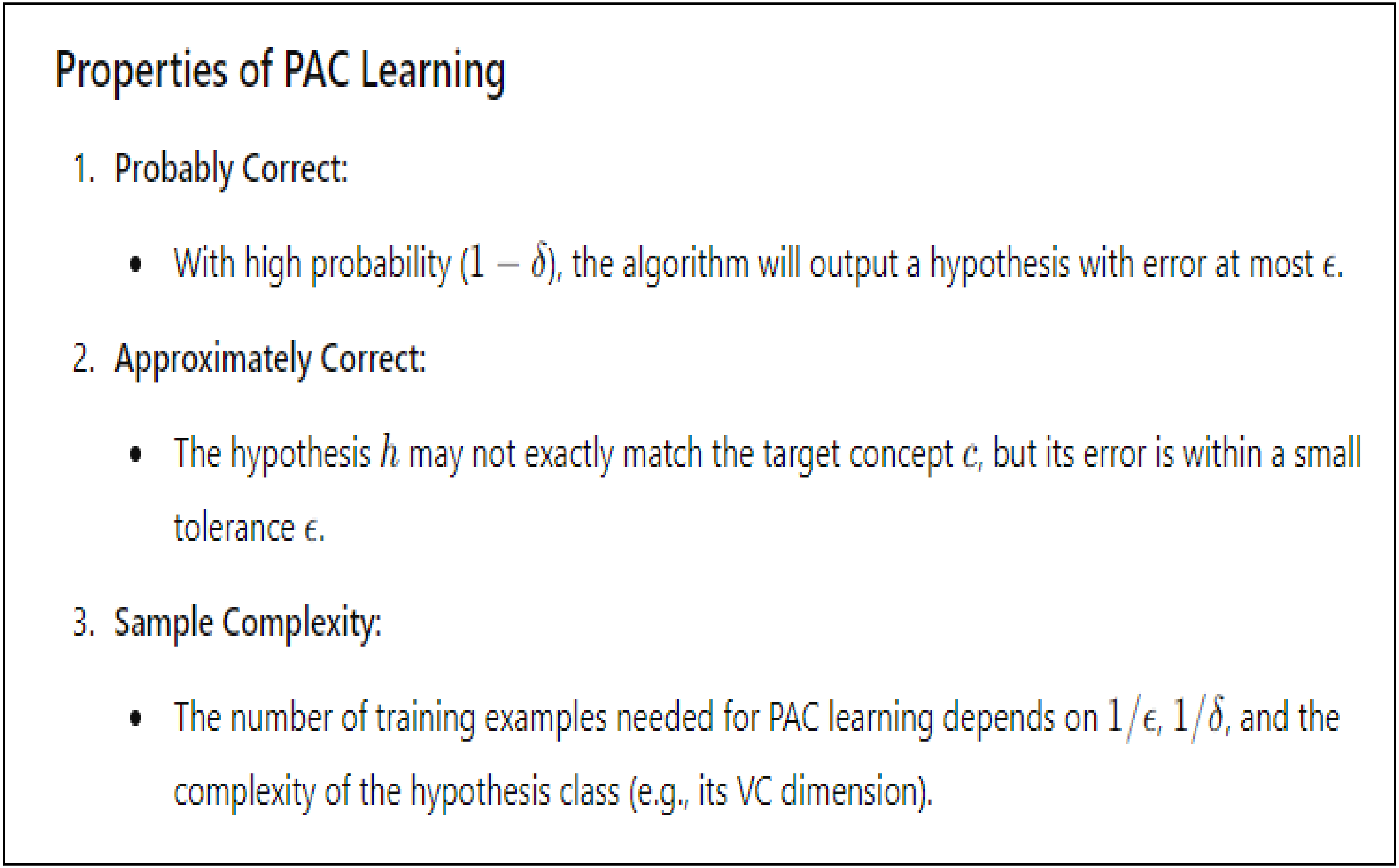
$$\Pr[\mathrm{err}_D(h) \leq \epsilon] \geq 1 - \delta$$

This holds after a number of training examples that is polynomial in $|H|$, $1/\epsilon$, and $1/\delta$.



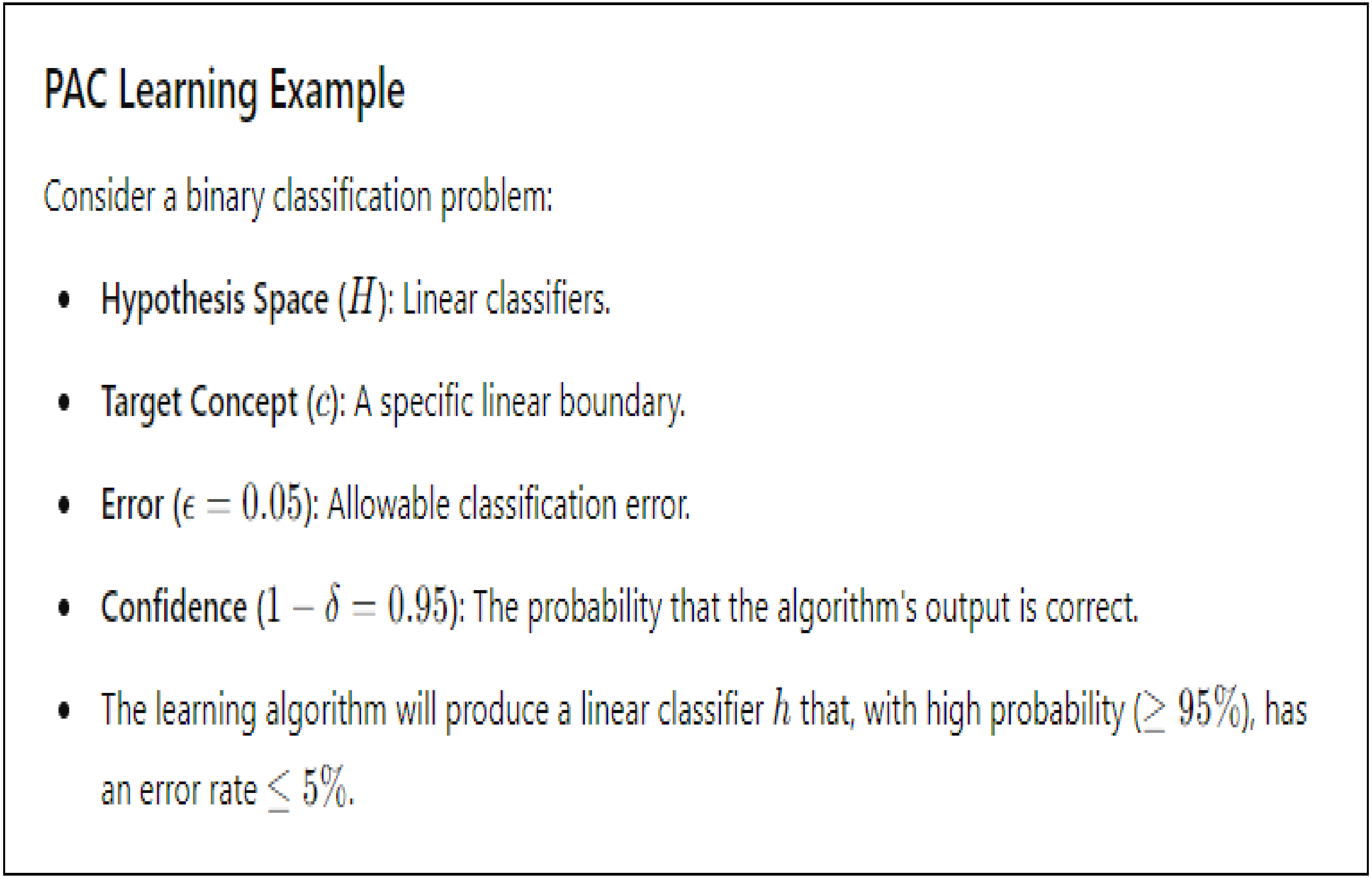
Requirements for PAC Learnability

1. Finite Hypothesis Space:
   - If H is finite, then PAC learning is straightforward as it involves searching for a hypothesis that fits the training data well. The size of H determines the number of samples needed.
2. VC Dimension:
   - For infinite hypothesis spaces, the **Vapnik-Chervonenkis (VC) dimension** is used to measure the capacity of H. A class H is PAC-learnable if its VC dimension is finite.



Properties of PAC Learning

1. **Probably Correct:**
   - With high probability $(1 - \delta)$, the algorithm will output a hypothesis with error at most ϵ.
2. **Approximately Correct:**
   - The hypothesis h may not exactly match the target concept c, but its error is within a small tolerance ϵ.
3. **Sample Complexity:**
   - The number of training examples needed for PAC learning depends on $1/\epsilon$, $1/\delta$, and the complexity of the hypothesis class (e.g., its VC dimension).

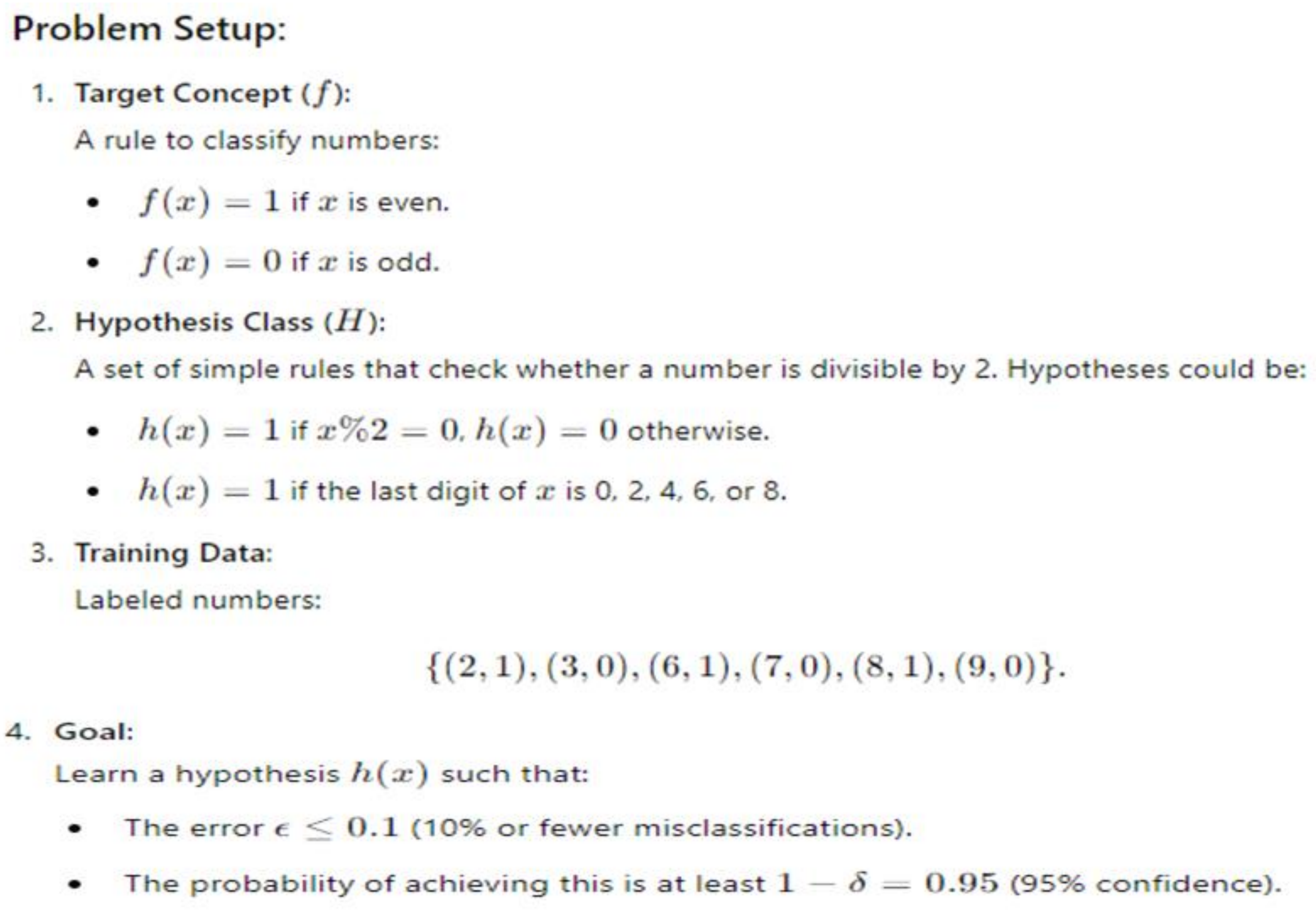


PAC Learning Example

Consider a binary classification problem:

- **Hypothesis Space (H)**: Linear classifiers.
- **Target Concept (c)**: A specific linear boundary.
- **Error ($\epsilon = 0.05$)**: Allowable classification error.
- **Confidence ($1 - \delta = 0.95$)**: The probability that the algorithm's output is correct.
- The learning algorithm will produce a linear classifier h that, with high probability ($\geq 95\%$), has an error rate $\leq 5\%$.

PAC Learning : Example - Learning to Classify Even and Odd Numbers



Problem Setup:

1. **Target Concept (f):**

   A rule to classify numbers:

   - $f(x) = 1$ if x is even.
   - $f(x) = 0$ if x is odd.

2. **Hypothesis Class (H):**

   A set of simple rules that check whether a number is divisible by 2. Hypotheses could be:

   - $h(x) = 1$ if $x\%2 = 0$, $h(x) = 0$ otherwise.
   - $h(x) = 1$ if the last digit of x is 0, 2, 4, 6, or 8.

3. **Training Data:**

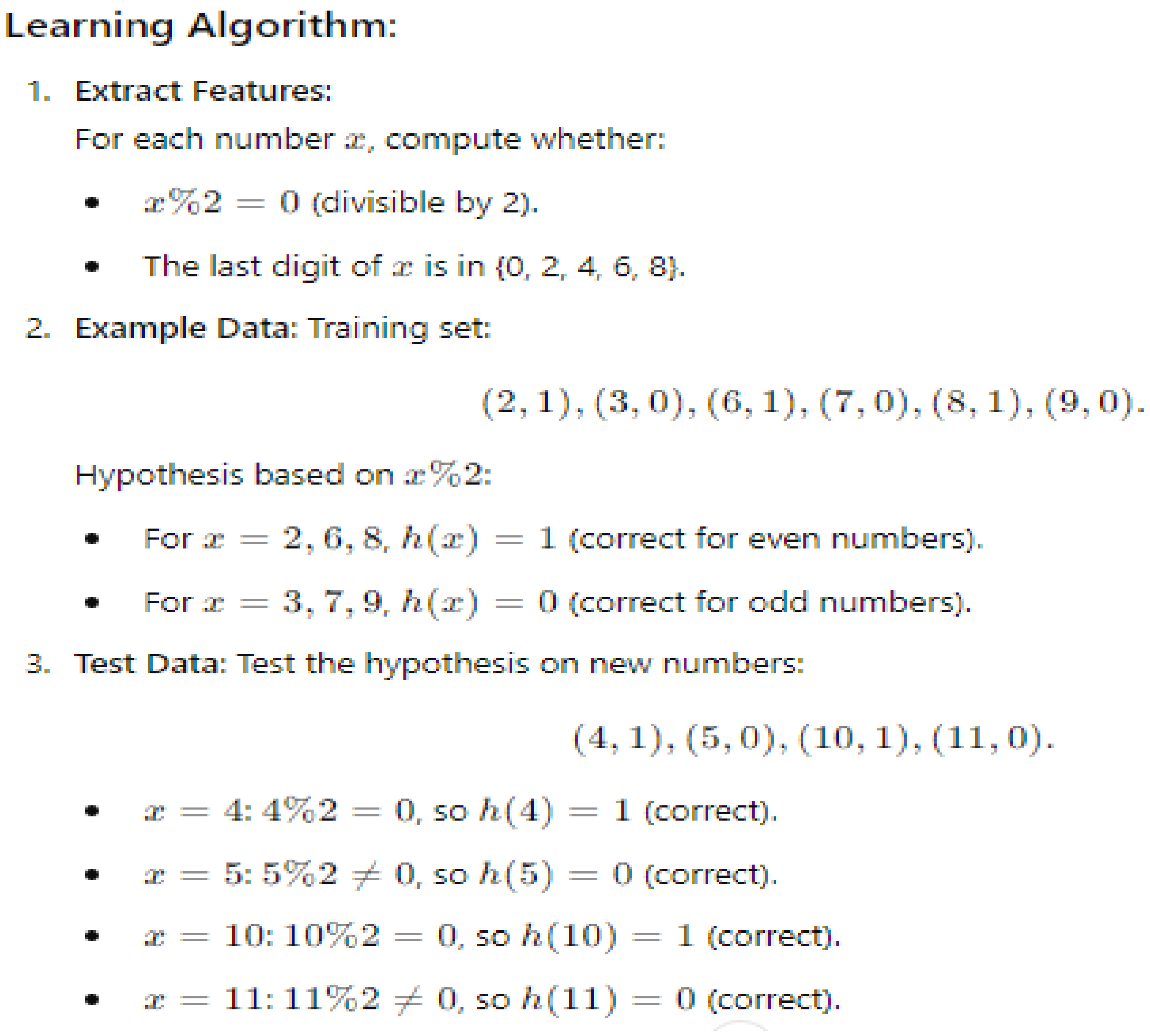
   Labeled numbers:

   $$\{(2,1),(3,0),(6,1),(7,0),(8,1),(9,0)\}.$$

4. **Goal:**

   Learn a hypothesis $h(x)$ such that:

   - The error $\epsilon \leq 0.1$ (10% or fewer misclassifications).
   - The probability of achieving this is at least $1 - \delta = 0.95$ (95% confidence).



Learning Algorithm:

1. **Extract Features:**

   For each number x, compute whether:

   - $x\%2 = 0$ (divisible by 2).
   - The last digit of x is in {0, 2, 4, 6, 8}.

2. **Example Data:** Training set:

   $$(2,1),(3,0),(6,1),(7,0),(8,1),(9,0).$$

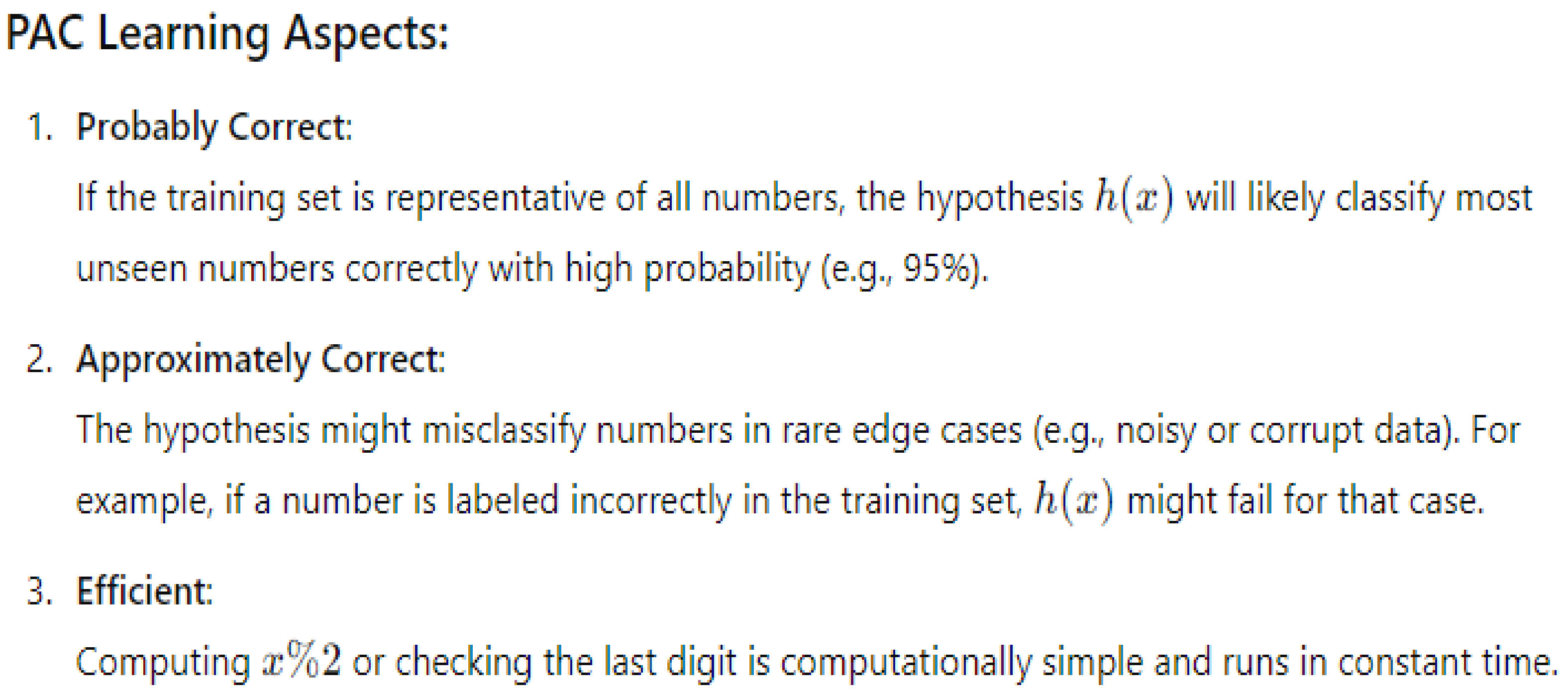
   Hypothesis based on $x\%2$:

   - For $x = 2, 6, 8$, $h(x) = 1$ (correct for even numbers).
   - For $x = 3, 7, 9$, $h(x) = 0$ (correct for odd numbers).

3. **Test Data:** Test the hypothesis on new numbers:

   $$(4,1),(5,0),(10,1),(11,0).$$

   - $x = 4$: $4\%2 = 0$, so $h(4) = 1$ (correct).
   - $x = 5$: $5\%2 \neq 0$, so $h(5) = 0$ (correct).
   - $x = 10$: $10\%2 = 0$, so $h(10) = 1$ (correct).
   - $x = 11$: $11\%2 \neq 0$, so $h(11) = 0$ (correct).



PAC Learning Aspects:

1. **Probably Correct:**

   If the training set is representative of all numbers, the hypothesis $h(x)$ will likely classify most unseen numbers correctly with high probability (e.g., 95%).

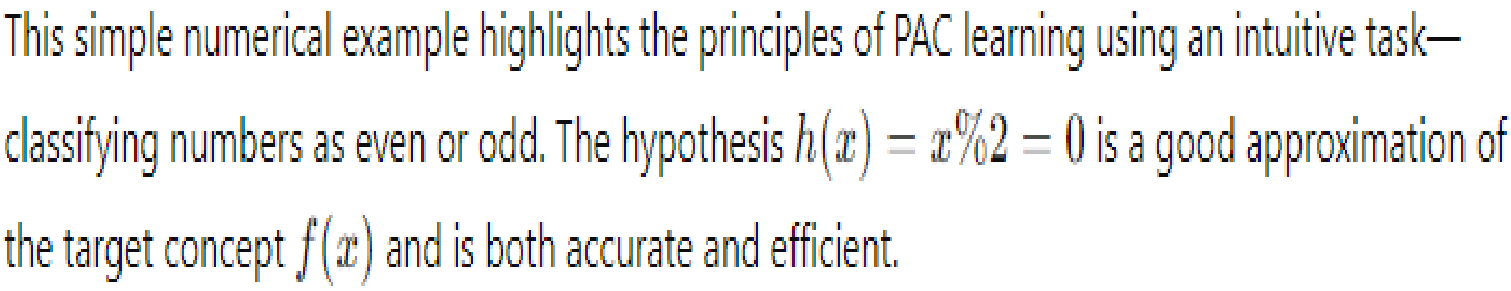
2. **Approximately Correct:**

   The hypothesis might misclassify numbers in rare edge cases (e.g., noisy or corrupt data). For example, if a number is labeled incorrectly in the training set, $h(x)$ might fail for that case.

3. **Efficient:**

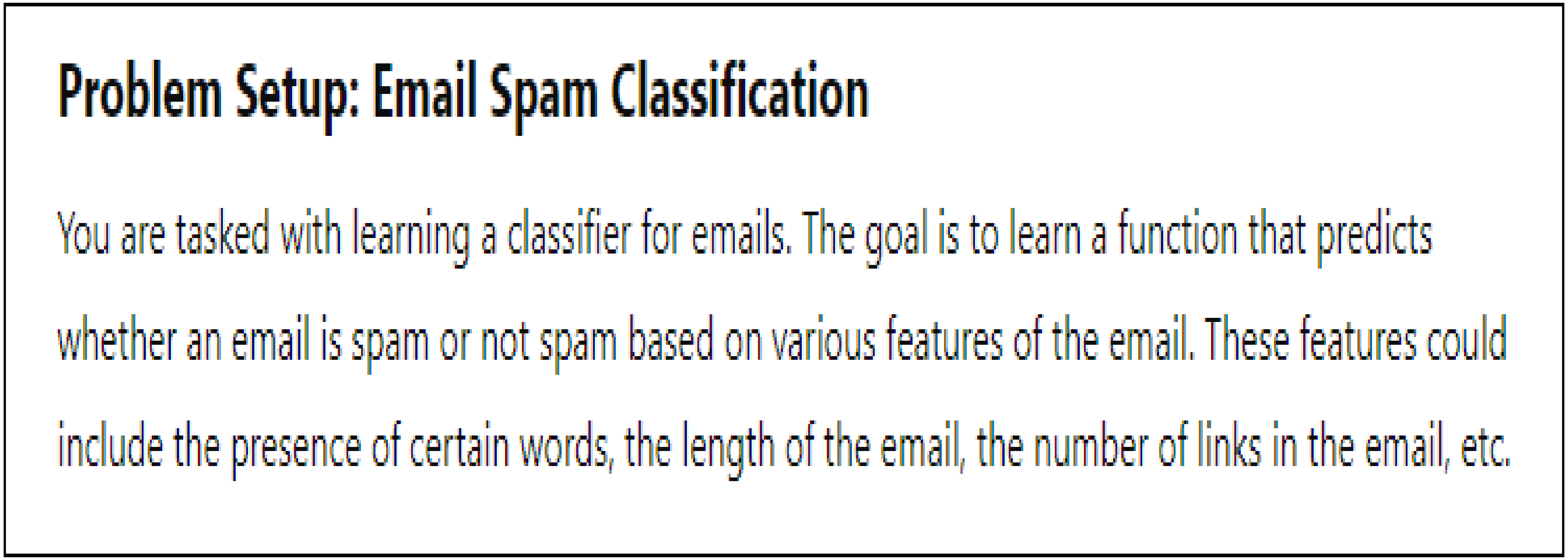
   Computing $x\%2$ or checking the last digit is computationally simple and runs in constant time.

Conclusion



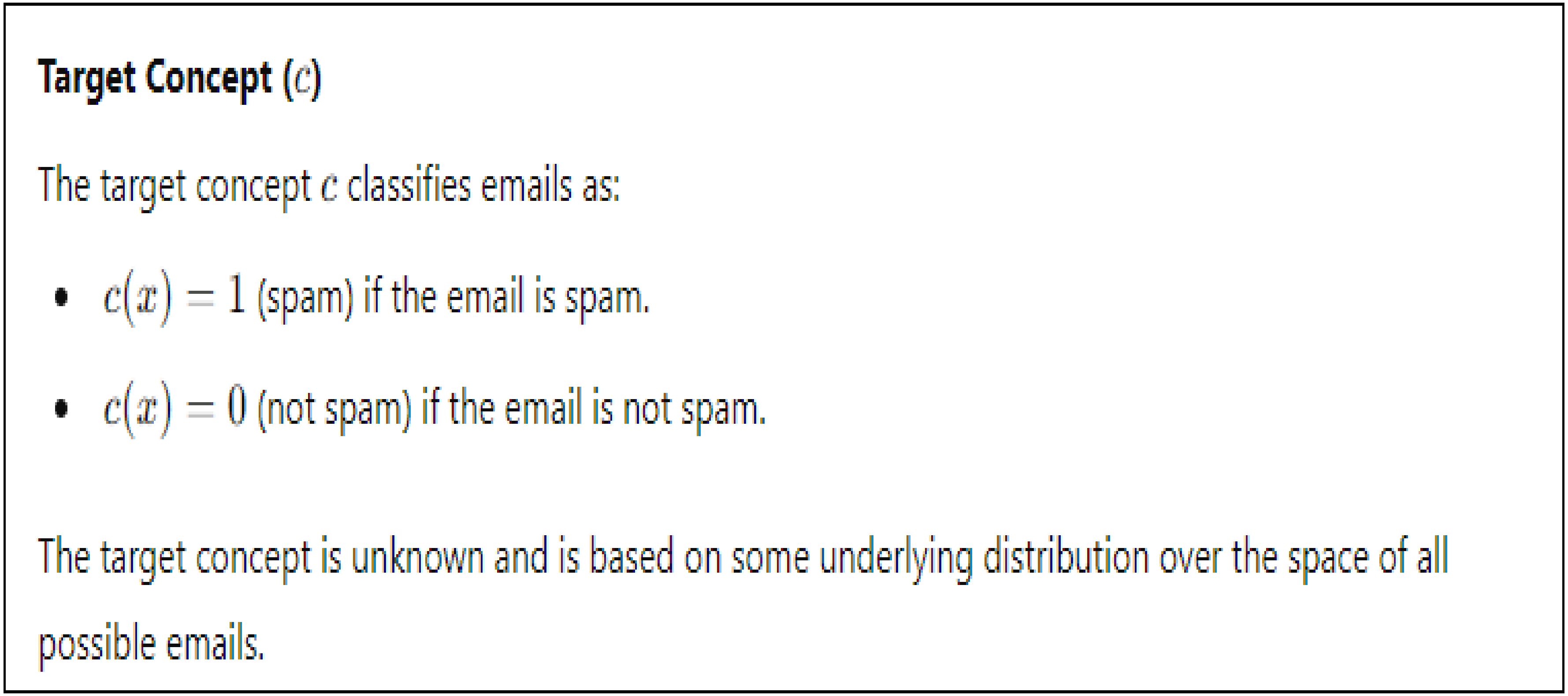
This simple numerical example highlights the principles of PAC learning using an intuitive task—classifying numbers as even or odd. The hypothesis $h(x) = x\%2 = 0$ is a good approximation of the target concept $f(x)$ and is both accurate and efficient.

Another Example



Problem Setup: Email Spam Classification

You are tasked with learning a classifier for emails. The goal is to learn a function that predicts whether an email is spam or not spam based on various features of the email. These features could include the presence of certain words, the length of the email, the number of links in the email, etc.

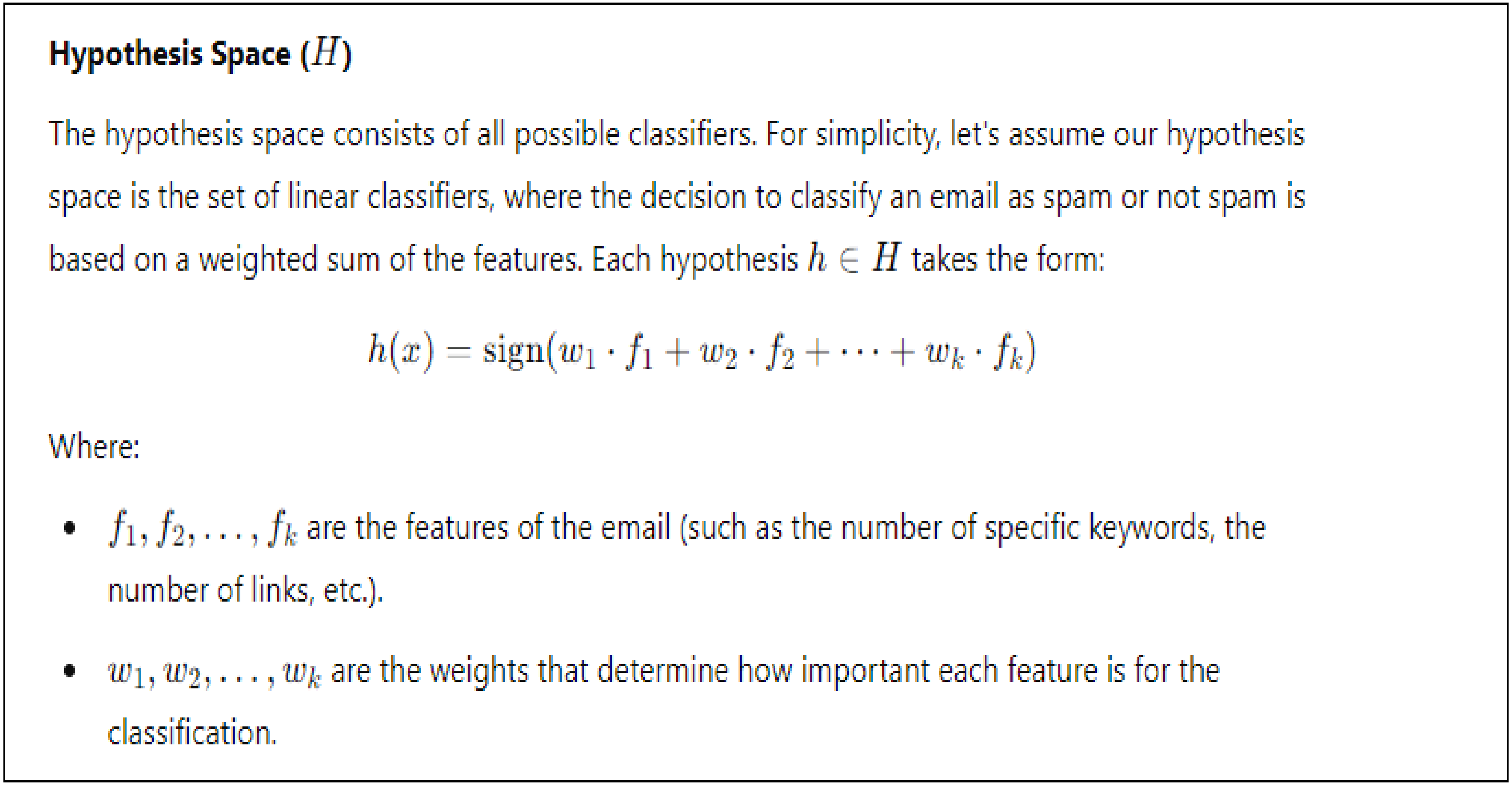


Target Concept (c)

The target concept c classifies emails as:

- $c(x) = 1$ (spam) if the email is spam.
- $c(x) = 0$ (not spam) if the email is not spam.

The target concept is unknown and is based on some underlying distribution over the space of all possible emails.



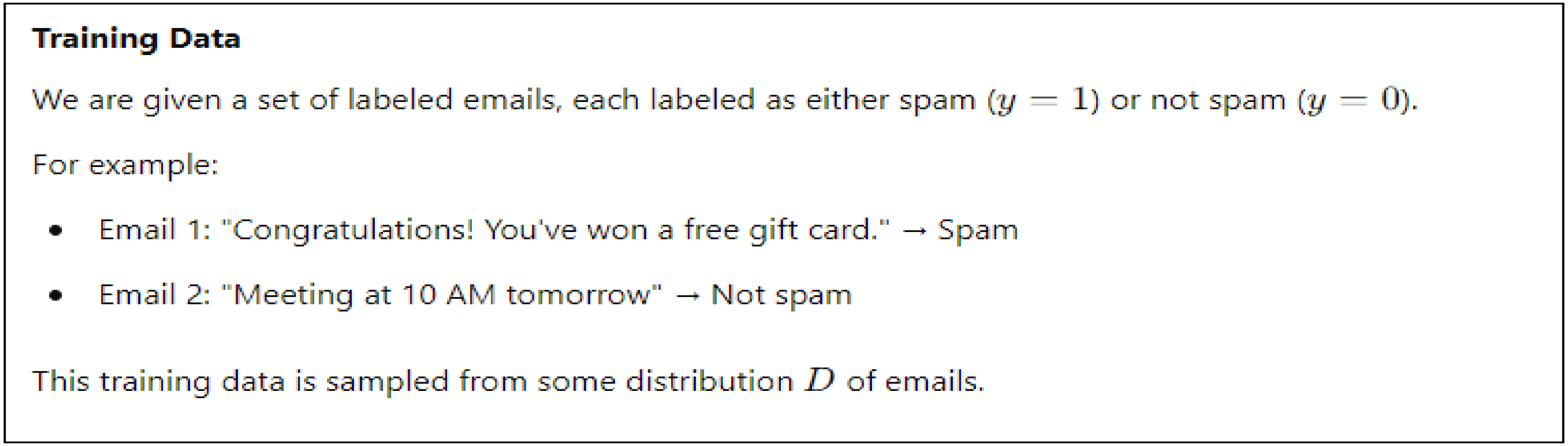
Hypothesis Space (H)

The hypothesis space consists of all possible classifiers. For simplicity, let's assume our hypothesis space is the set of linear classifiers, where the decision to classify an email as spam or not spam is based on a weighted sum of the features. Each hypothesis $h \in H$ takes the form:

$$h(x) = \text{sign}(w_1 \cdot f_1 + w_2 \cdot f_2 + \cdots + w_k \cdot f_k)$$

Where:

- $f_1, f_2, \ldots, f_k$ are the features of the email (such as the number of specific keywords, the number of links, etc.).
- $w_1, w_2, \ldots, w_k$ are the weights that determine how important each feature is for the classification.



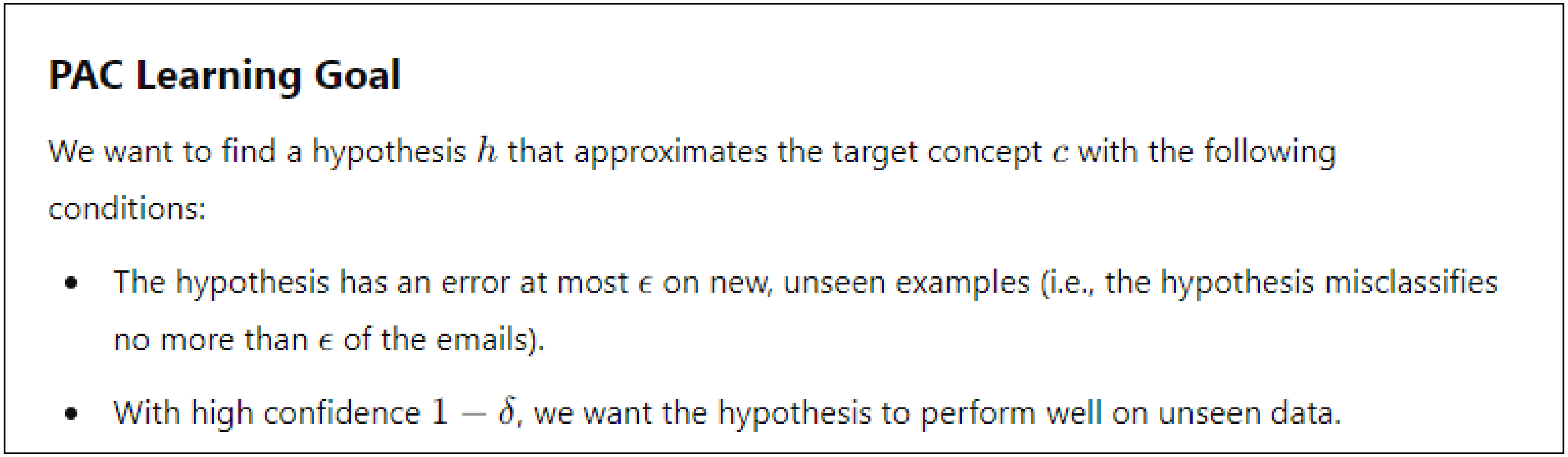
Training Data

We are given a set of labeled emails, each labeled as either spam ($y = 1$) or not spam ($y = 0$).

For example:

- Email 1: "Congratulations! You've won a free gift card." → Spam
- Email 2: "Meeting at 10 AM tomorrow" → Not spam

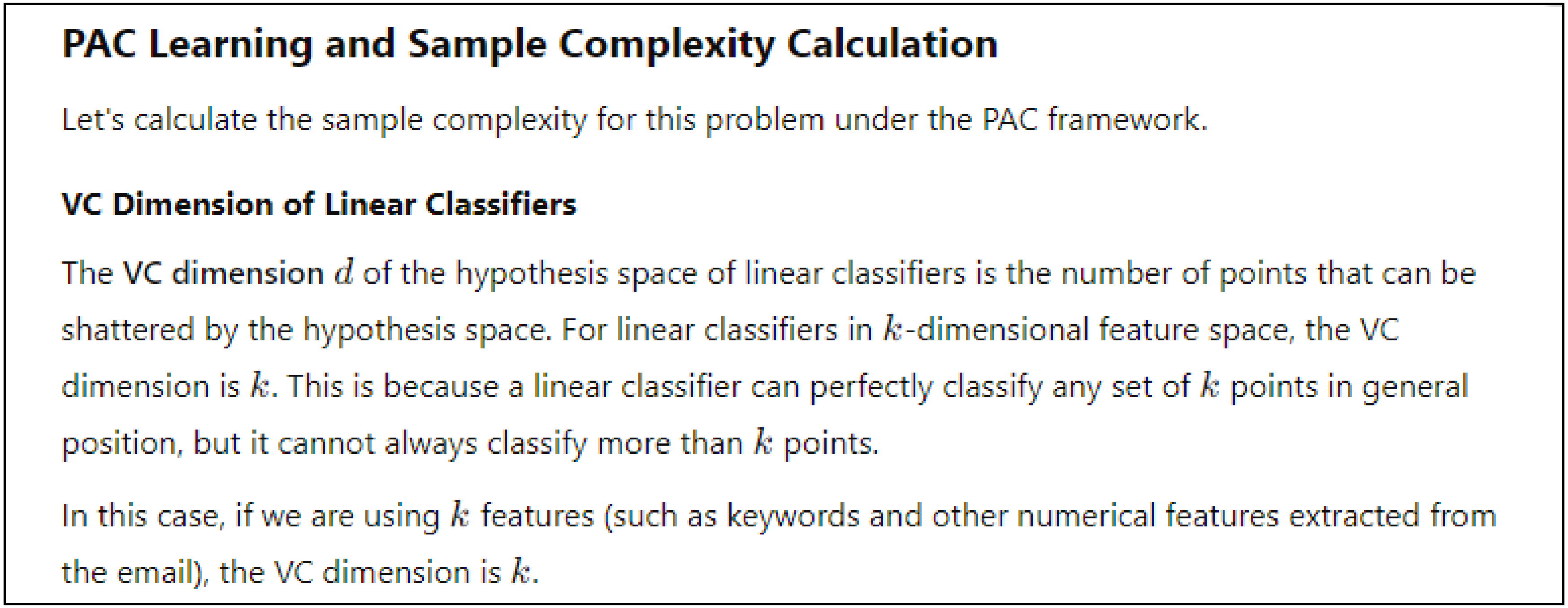
This training data is sampled from some distribution D of emails.



PAC Learning Goal

We want to find a hypothesis h that approximates the target concept c with the following conditions:

- The hypothesis has an error at most ϵ on new, unseen examples (i.e., the hypothesis misclassifies no more than ϵ of the emails).
- With high confidence $1 - \delta$, we want the hypothesis to perform well on unseen data.



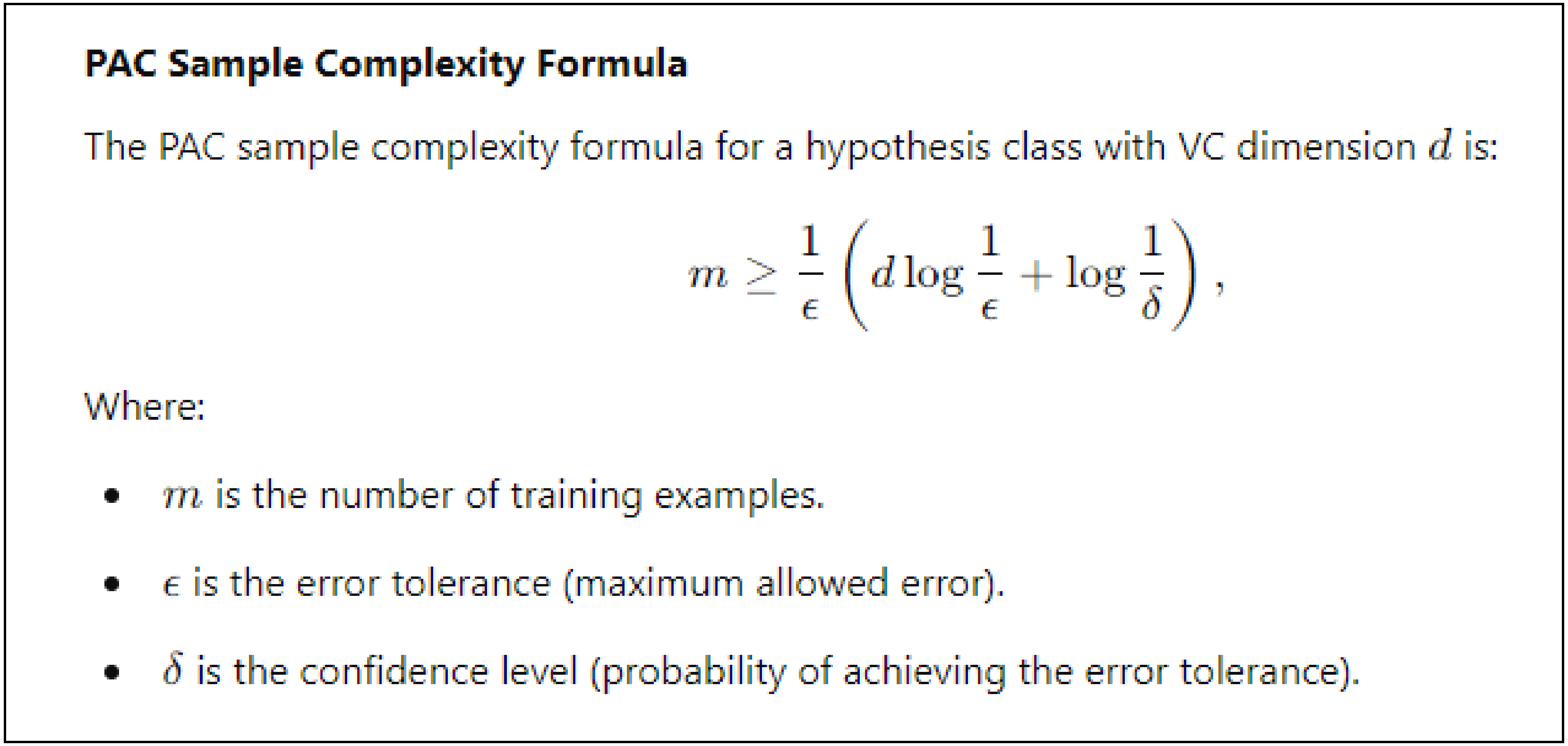
PAC Learning and Sample Complexity Calculation

Let's calculate the sample complexity for this problem under the PAC framework.

VC Dimension of Linear Classifiers

The **VC dimension** d of the hypothesis space of linear classifiers is the number of points that can be shattered by the hypothesis space. For linear classifiers in k-dimensional feature space, the VC dimension is k. This is because a linear classifier can perfectly classify any set of k points in general position, but it cannot always classify more than k points.

In this case, if we are using k features (such as keywords and other numerical features extracted from the email), the VC dimension is k.



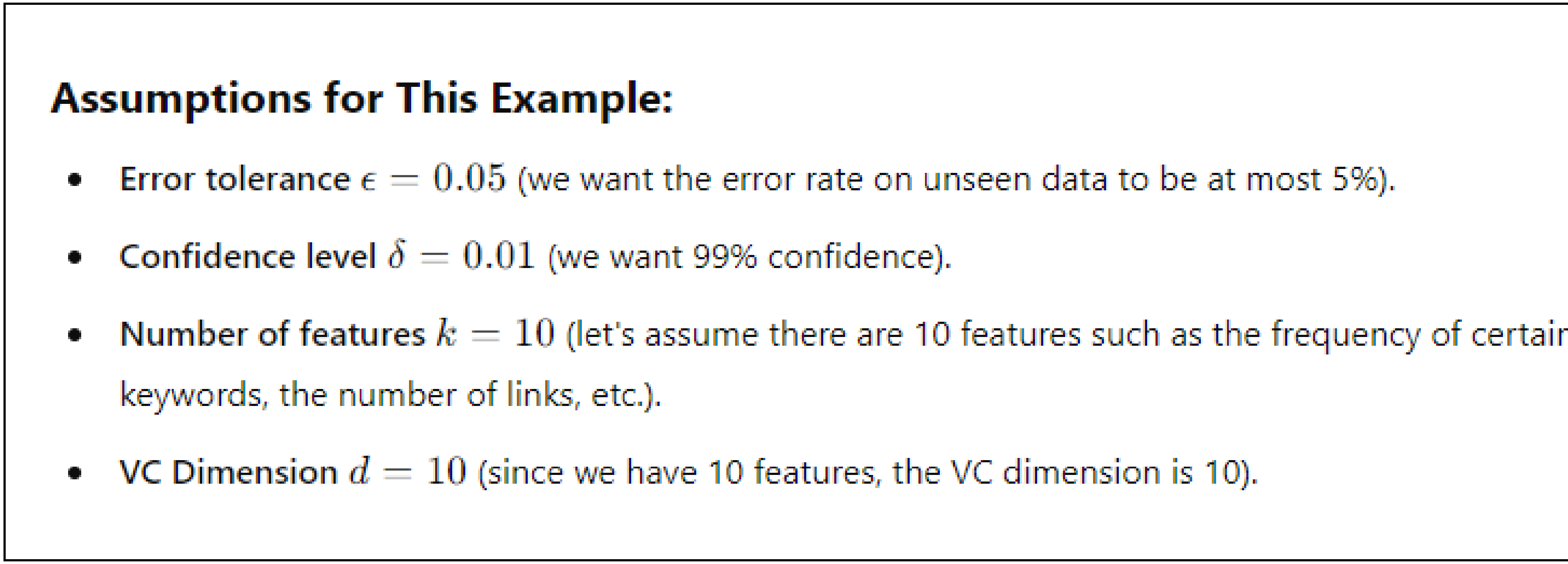
PAC Sample Complexity Formula

The PAC sample complexity formula for a hypothesis class with VC dimension d is:

$$m \geq \frac{1}{\epsilon}\left(d\log\frac{1}{\epsilon} + \log\frac{1}{\delta}\right),$$

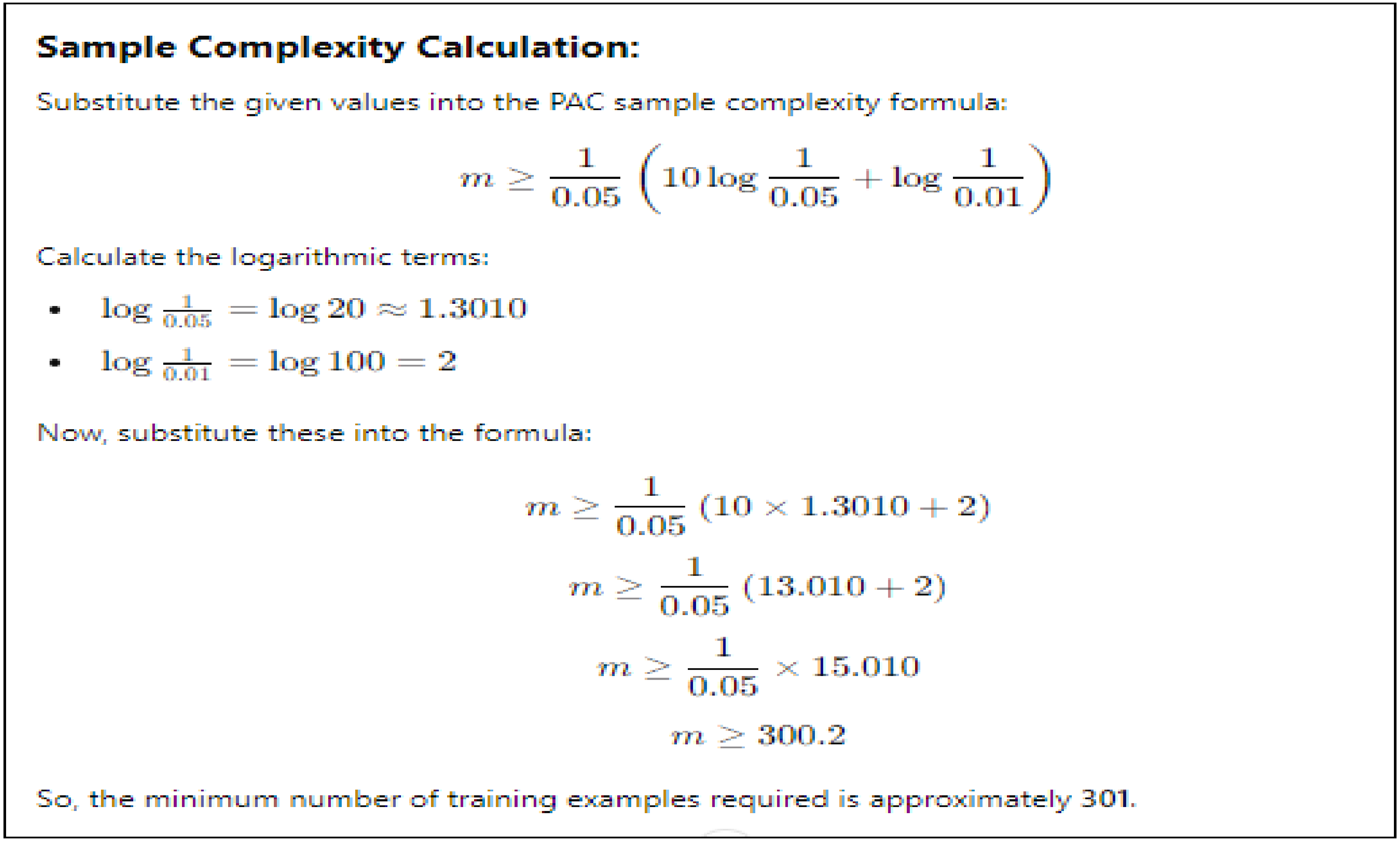
Where:

- m is the number of training examples.
- ϵ is the error tolerance (maximum allowed error).
- δ is the confidence level (probability of achieving the error tolerance).



Assumptions for This Example:

- Error tolerance $\epsilon = 0.05$ (we want the error rate on unseen data to be at most 5%).
- Confidence level $\delta = 0.01$ (we want 99% confidence).
- Number of features $k = 10$ (let's assume there are 10 features such as the frequency of certai keywords, the number of links, etc.).
- VC Dimension $d = 10$ (since we have 10 features, the VC dimension is 10).



Sample Complexity Calculation:

Substitute the given values into the PAC sample complexity formula:

$$m \geq \frac{1}{0.05}\left(10\log\frac{1}{0.05} + \log\frac{1}{0.01}\right)$$

Calculate the logarithmic terms:

- $\log\frac{1}{0.05} = \log 20 \approx 1.3010$
- $\log\frac{1}{0.01} = \log 100 = 2$

Now, substitute these into the formula:

$$m \geq \frac{1}{0.05}(10 \times 1.3010 + 2)$$

$$m \geq \frac{1}{0.05}(13.010 + 2)$$

$$m \geq \frac{1}{0.05} \times 15.010$$

$$m \geq 300.2$$

So, the minimum number of training examples required is approximately 301.

To ensure that our spam classifier has an error of at most 5% with 99% confidence, we need to gather at least **301 labeled training examples**.

PROBLEMS WITH PAC

- The PAC Learning framework has 2 disadvantages:
  - It can lead to weak bounds
  - Sample Complexity bound cannot be established for infinite hypothesis spaces

- We introduce the VC dimension for dealing with these problems (particularly the second one)