Basics of Learning Theory

Module 3

INTRODUCTION TO LEARNING AND ITS TYPES

- The process of acquiring knowledge and expertise through study, experience, or being taught is called as learning.
- Generally, humans learn in different ways. To make machines learn, we need to simulate the strategies of human learning in machines.
  - But, will the computers learn?
  - This question has been raised over many centuries by philosophers, mathematicians and logicians.
- First let us address the question - What sort of tasks can the computers learn?
  - This depends on the nature of problems that the computers can solve. There are two kinds of problems - well-posed and ill-posed. Computers can solve only well-posed problems, as these have well-defined specifications and have the following components inherent to it.

INTRODUCTION TO LEARNING AND ITS TYPES

- 1. Class of learning tasks (T)2. A measure of performance (P)3. A source of experience (E)
- The standard definition of learning proposed by Tom Mitchell is that a program can learn from E for the task T, and P improves with experience E. Let us formalize the concept of learning as follows:
- Let D be the input dataset with examples, (x1,y1),(x2,y2), ... ,(xn,yn) for n inputs.

INTRODUCTION TO LEARNING AND ITS TYPES

- Let the unknown target function be f: X -> Y, that maps the input space to output space.
- The objective of the learning program is to pick a function, g: x -> Y to approximate hypothesis f.
- All the possible formulae form a hypothesis space. In short, let H be the set of all formulae from which the learning algorithm chooses. The choice is good when the hypothesis g replicates f for all samples. This is shown in Figure 3.1.

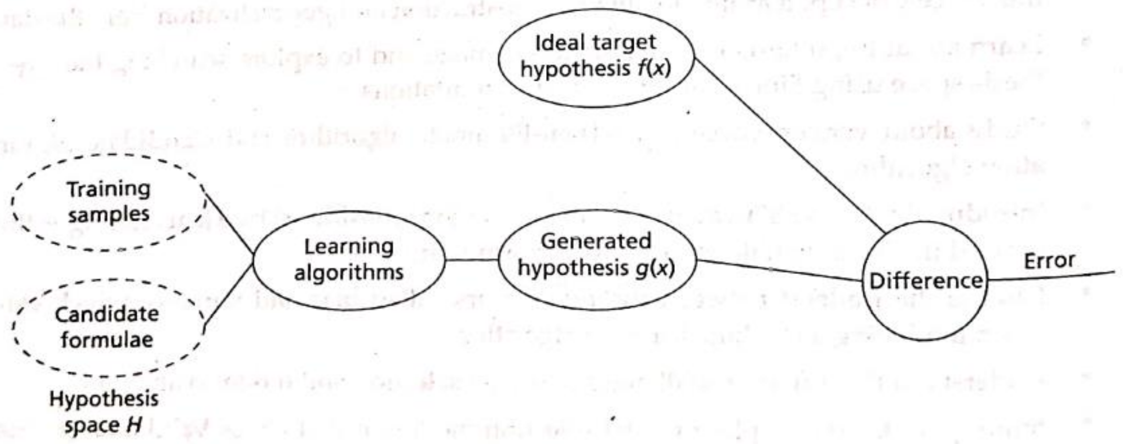


Figure 3.1: Learning Environment

INTRODUCTION TO LEARNING AND ITS TYPES

- It can be observed that training samples and target function are dependent on the given problem. The learning algorithm and hypothesis set are independent of the given problem. Thus learning model is informally the hypothesis set and learning algorithm. Thus, learning model can be stated as follows:
  - Learning Model = Hypothesis Set + Learning Algorithm
- Let us assume a problem of predicting a label for a given input data. Let D be the input dataset with both positive and negative examples. Let y be the output with class 0 or 1. The simple learning model can be given as:

$$h(x) = sign\left(\left(\sum_{i=1}^{D} x_i w_i\right) + b\right)$$

Learning Types

- There are different types of learning. Some of the different learning methods are as follows:
  - Learn by memorization or learn by repetition also called as rote learning is done by memorizing without understanding the logic or concept. Although rote learning is basically learning by repetition, in machine learning perspective, the learning occurs by simply comparing with the existing knowledge for the same input data and producing the output if present.
  - Learn by examples also called as learn by experience or previous knowledge acquired-at some time, is like finding an analogy, which means performing inductive learning from observations that formulate a general concept. Here, the learner learns by inferring a general rule from the set of observations or examples. Therefore, inductive learning is also called as discovery learning.

Learning Types

- Learn by being taught by an expert or a teacher, generally called as passive learning. However, there is a special kind of learning called active learning where the learner can interactively query a teacher/expert to label unlabelled data instances with the desired outputs.
- Learning by critical thinking, also called as deductive learning, deduces new facts or conclusion from related known facts and information.
- Self learning, also called as reinforcement learning, is a self-directed learning that normally learns from mistakes punishments and rewards.

Learning Types

- Learning to solve problems is a type of cognitive learning where learning happens in the mind and is possible by devising a methodology to achieve a goal. Here, the learner initially is not aware of the solution or the way to achieve the goal but only knows the goal. The learning happens either directly from the initial state by following the steps to achieve the goal or indirectly by inferring the behavior.
- Learning by generalizing explanations, also called as explanation-based learning (EBL),is another learning method that exploits domain knowledge from experts to improve the accuracy of learned concepts by supervised learning.

INTRODUCTION TO COMPUTATION LEARNING THEORY

- There are many questions that have been raised by mathematicians and logicians over the time taken by computers to learn. Some of the questions are as follows:
  - How can a learning system predict an unseen instance?
  - How do the hypothesis h is close to f, when hypothesis f itself is unknown?
  - How many samples are required?
  - Can we measure the performance of a learning system?
  - Is the solution obtained local or global?

INTRODUCTION TO COMPUTATION LEARNING THEORY

- These questions are the basis of a field called 'Computational Learning Theory' or in short(COLT).
- It is a specialized field of study of machine learning. COLT deals with formal methods used for learning systems. It deals with frameworks for quantifying learning tasks and learning algorithms.
- It provides a fundamental basis for study of machine learning.
- It deals with Probably Approximate Learning (PAC) and Vapnik-Chervonenkis (VC) dimensions. The focus of PAC is the quantification of the computational difficulty of learning tasks and algorithms and the computation capacity quantification is the focus of VC dimension.

Vapnik–Chervonenkis theory, the Vapnik–Chervonenkis (VC) dimension is a measure of the size (capacity, complexity, expressive power, richness, or flexibility) of a class of sets. The notion can be extended to classes of binary functions

DESIGN OF A LEARNING SYSTEM

- A system that is built around a learning algorithm is called a learning system.
- The design of systems focuses on these steps:
    - Choosing a training experience
    - Choosing a target function
    - Representation of a target function
    - Function approximation

Training Experience

- Let us consider designing of a chess game.
  - In direct experience, individual board states and correct moves of the chess game are given directly.
  - In indirect system, the move sequences and results are only given.
  - The training experience also depends on the presence of a supervisor who can label all valid moves for a board state.
  - In the absence of a supervisor, the game agent plays against itself and learns the good moves, if the training samples cover all scenarios, or in other words, distributed enough for performance computation.
  - If the training samples and testing samples have the same distribution, the results would be good.

Determine the Target Function

- The next step is the determination of a target function.
  - In this step, the type of knowledge that needs to be learnt is determined.
  - In direct experience, a board move is selected and is determined whether it is a good move or not against all other moves.
  - If it is the best move, then it is chosen as:

    B-> M, where, B and M are legal moves.
  - In indirect experience, all legal moves are accepted and a score is generated for each. The move with largest score is then chosen and executed.

Determine the Target Function Representation

- The representation of knowledge may be a table, collection of rules or a neural network. The linear combination of these factors can be coined as:

$$V = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Choosing an Approximation Algorithm for the Target Function

- The focus is to choose weights and fit the given training samples effectively. The aim is to reduce the error given as:

$$E \equiv \sum_{\substack{Training \\ Samples}} \left[V_{train}(b) - \hat{V}(b) \right]^2$$

- Here, b is the sample and V(b) is the predicted hypothesis.
- The approximation is carried out as:
  - Computing the error as the difference between trained and expected hypothesis. Let error be error(b).
  - Then, for every board feature x_i, the weights are updated as:

$$w_i' = w_i + \mu \times error(b) \times x_i$$

Here, μ is the constant that moderates the size of the weight update.

INTRODUCTION TO CONCEPT LEARNING

- Concept learning is a learning strategy of acquiring abstract knowledge or inferring a general concept or deriving a category from the given training samples.
- It is a process of abstraction and generalization from the data.
- Concept learning helps to classify an object that has a set of common, relevant features.
- Thus, it helps a learner compare and contrast categories based on the similarity and association of positive and negative instances in the training data to classify an object.
- The learner tries to simplify by observing the common features from the training samples and then apply this simplified model to the future samples.
- This task is also known as learning from experience.

INTRODUCTION TO CONCEPT LEARNING

- Concept learning requires three things:
  - Input - Training dataset which is a set of training instances, each labeled with the name of a concept or category to which it belongs. Use this past experience to train and build the model.
  - Output - Target concept or Target function f. It is a mapping function f(x) from input x to output y. It is to determine the specific features or common features to identify an object.
    - In other words, it is to find the hypothesis to determine the target concept. For e.g., the specific set of features to identify an elephant from all animals.
  - Test - New instances to test the learned model.

Formally, Concept learning is defined as-"Given a set of hypotheses, the learner searches through the hypothesis space to identify the best hypothesis that matches the target concept".

INTRODUCTION TO CONCEPT LEARNING

- Consider the following set of training instances shown in Table 3.1.

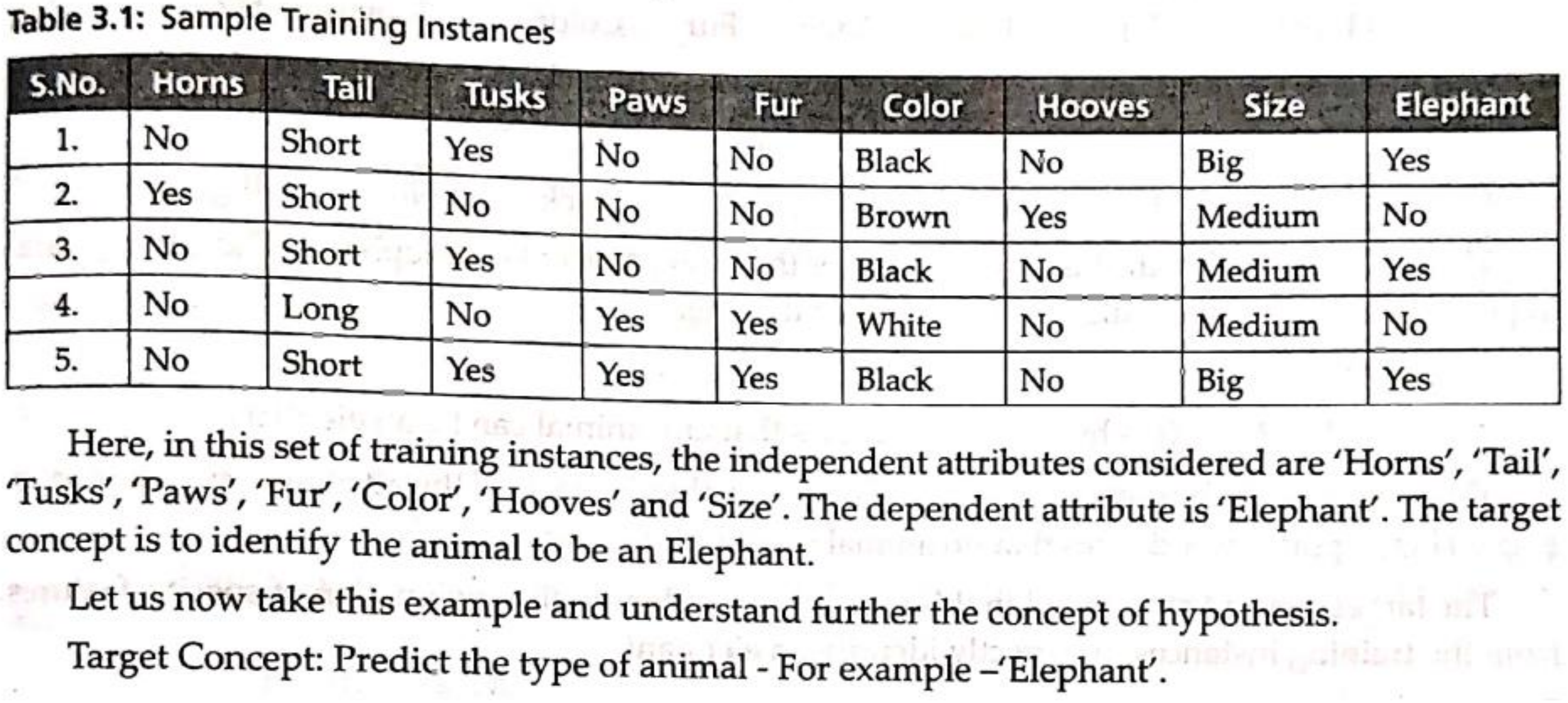


Table 3.1: Sample Training Instances

S.No.	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1.	No	Short	Yes	No	No	Black	No	Big	Yes
2.	Yes	Short	No	No	No	Brown	Yes	Medium	No
3.	No	Short	Yes	No	No	Black	No	Medium	Yes
4.	No	Long	No	Yes	Yes	White	No	Medium	No
5.	No	Short	Yes	Yes	Yes	Black	No	Big	Yes

Here, in this set of training instances, the independent attributes considered are 'Horns', 'Tail', 'Tusks', 'Paws', 'Fur', 'Color', 'Hooves' and 'Size'. The dependent attribute is 'Elephant'. The target concept is to identify the animal to be an Elephant.

Let us now take this example and understand further the concept of hypothesis.

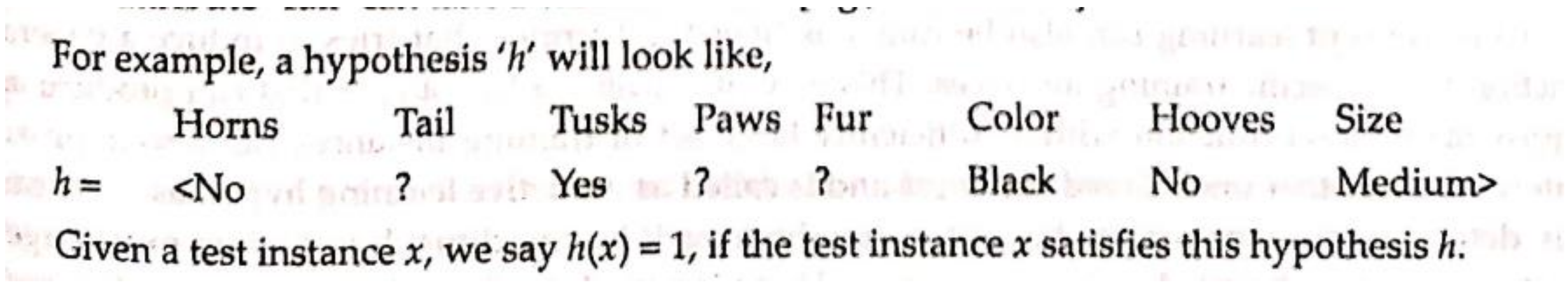
Target Concept: Predict the type of animal - For example –'Elephant'.

Representation of a Hypothesis

- A hypothesis 'h' approximates a target function 'f' to represent the relationship between the independent attributes and the dependent attribute of the training instances.
- The hypothesis is the predicted approximate model that best maps the inputs to outputs. Each hypothesis is represented as a conjunction of attribute conditions in the antecedent part(Pronoun).
- For example, (Tail = Short) ^ (Color = Black) ....
- The set of hypothesis in the search space is called as hypotheses.
- Hypotheses are the plural form of hypothesis. Generally 'H' is used to represent the hypotheses and 'h' is used to represent a candidate hypothesis.

Representation of a Hypothesis

- Each attribute condition is the constraint on the attribute which is represented as attribute-value pair. In the antecedent of an attribute condition of a hypothesis, each attribute can take value as either '?’ or 'Ø' or can hold a single value.
  - "?" denotes that the attribute can take any value [e.g., Color =?]·
  - "Ø" denotes that the attribute cannot take any value, i.e., it represents a null value[e.g., Horns = Ø]·
  - Single value denotes a specific single value from acceptable values of the attribute, i.e., the attribute 'Tail' can take a value as 'short' [e.g., Tail = Short]

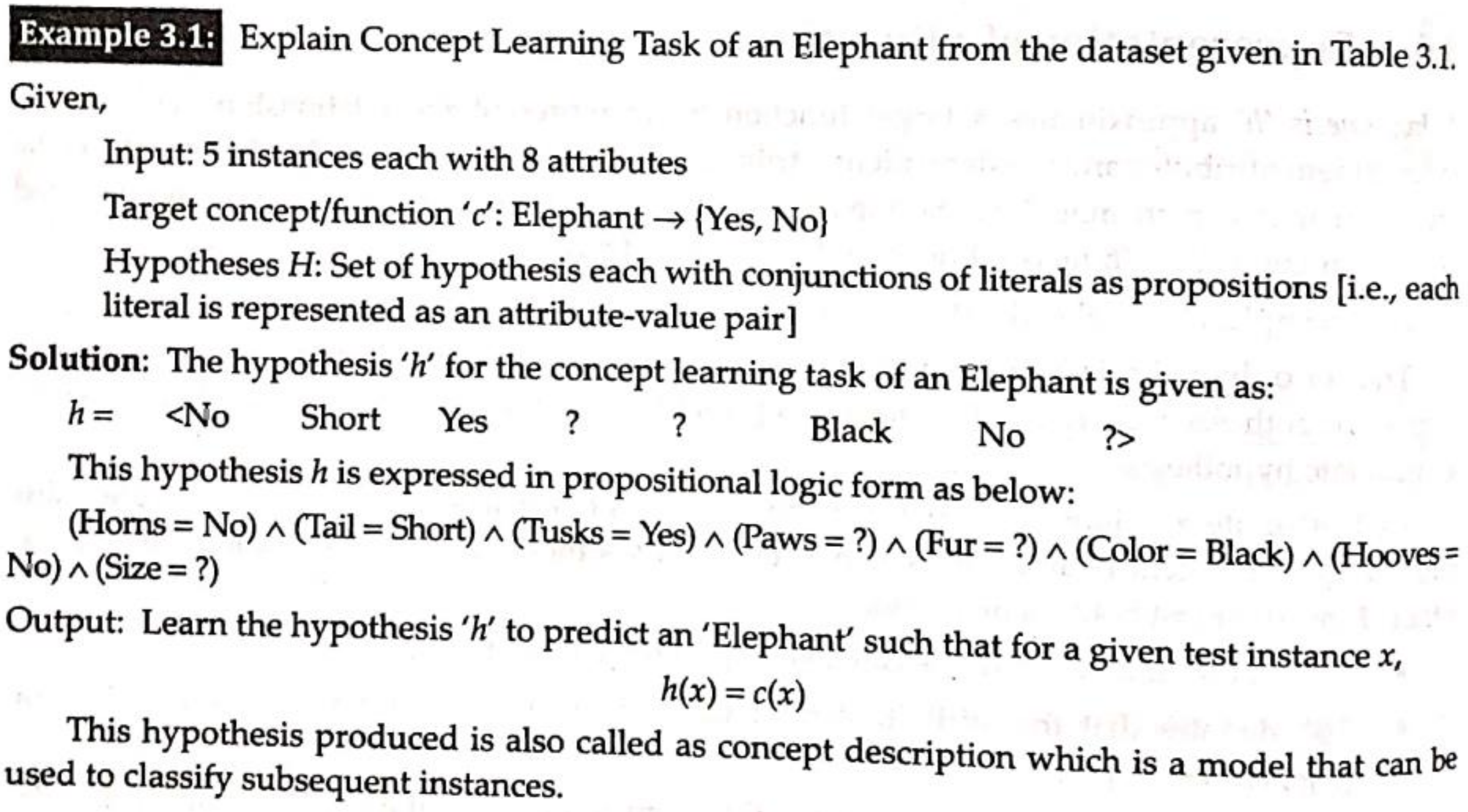


For example, a hypothesis *‘h’* will look like,

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
$h =$	<No	?	Yes	?	?	Black	No	Medium>

Given a test instance x, we say $h(x) = 1$, if the test instance x satisfies this hypothesis h.

Example



Example 3.1: Explain Concept Learning Task of an Elephant from the dataset given in Table 3.1.

Given,

Input: 5 instances each with 8 attributes

Target concept/function '*c*': Elephant → {Yes, No}

Hypotheses *H*: Set of hypothesis each with conjunctions of literals as propositions [i.e., each literal is represented as an attribute-value pair]

Solution: The hypothesis '*h*' for the concept learning task of an Elephant is given as:

$h =$ <No Short Yes ? ? Black No ?>

This hypothesis *h* is expressed in propositional logic form as below:

$(\text{Horns} = \text{No}) \wedge (\text{Tail} = \text{Short}) \wedge (\text{Tusks} = \text{Yes}) \wedge (\text{Paws} = ?) \wedge (\text{Fur} = ?) \wedge (\text{Color} = \text{Black}) \wedge (\text{Hooves} = \text{No}) \wedge (\text{Size} = ?)$

Output: Learn the hypothesis '*h*' to predict an 'Elephant' such that for a given test instance *x*,

$$h(x) = c(x)$$

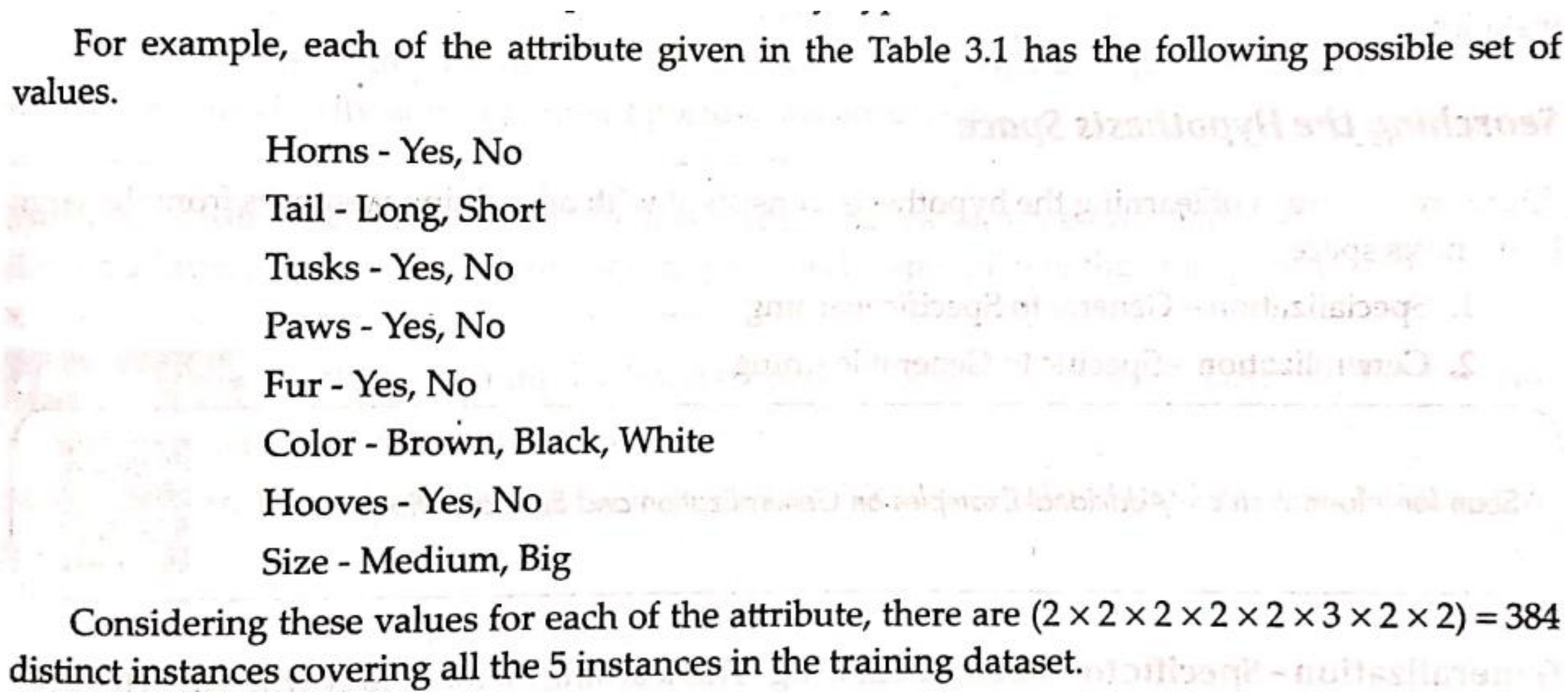
This hypothesis produced is also called as concept description which is a model that can be used to classify subsequent instances.

Hypothesis Space

- Hypothesis space is the set of all possible hypotheses that approximates the target function f.
- In other words, the set of all possible approximations of the target function can be defined as hypothesis space.
- From this set of hypotheses in the hypothesis space, a machine learning algorithm would determine the best possible hypothesis that would best describe the target function or best fit the outputs.
- Generally, a hypothesis representation language represents a larger hypothesis space.
- Every machine learning algorithm would represent the hypothesis space in a different manner about the function that maps the input variables to output variables. For example, a regression algorithm represents the hypothesis space as a linear function whereas a decision tree algorithm represents the hypothesis space as a tree.

Hypothesis Space

- The subset of hypothesis space that is consistent with all-observed training instances is called as Version Space. Version space represents the only hypotheses that are used for the classification.



For example, each of the attribute given in the Table 3.1 has the following possible set of values.

Horns - Yes, No

Tail - Long, Short

Tusks - Yes, No

Paws - Yes, No

Fur - Yes, No

Color - Brown, Black, White

Hooves - Yes, No

Size - Medium, Big

Considering these values for each of the attribute, there are $(2 \times 2 \times 2 \times 2 \times 2 \times 3 \times 2 \times 2) = 384$ distinct instances covering all the 5 instances in the training dataset.

Heuristic Space Search

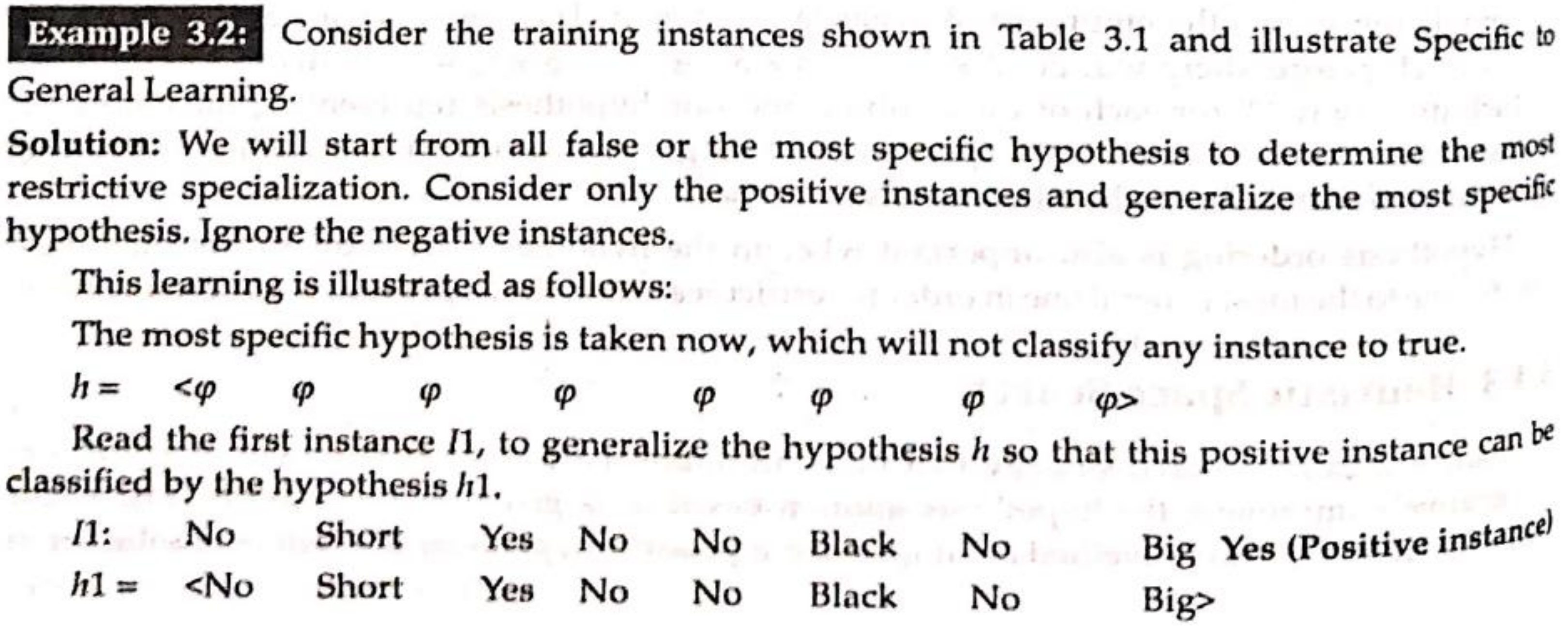
- Heuristic search is a search strategy that finds an optimized hypothesis/solution to a problem by iteratively improving the hypothesis/solution based on a given heuristic function or a cost measure.
- Heuristic search methods will generate a possible hypothesis that can be a solution in the hypothesis space or a path from the initial state.
- This hypothesis will be tested with the target function or the goal state to see if it is a real solution.
  - If the tested hypothesis is a real solution, then it will be selected.
  - This method generally increases the efficiency because it is guaranteed to find a better hypothesis but may not be the best hypothesis.
- It is useful for solving tough problems which could not solved by any other method.
- The typical example problem solved by heuristic search is the travelling salesman problem.
- Several commonly used heuristic search methods are hill climbing methods, constraint satisfaction problems, best-first search, simulated-annealing, A* algorithm, and genetic algorithms

Generalization and Specialization

- In order to understand about how we construct this concept hierarchy, let us apply this general principle of generalization/specialization relation.
- By generalization of the most specific hypothesis and by specialization of the most general hypothesis, the hypothesis space can be searched for an approximate hypothesis that matches all positive instances but does not match any negative instance.

Searching the Hypothesis Space

- There are two ways of learning the hypothesis, consistent with all training instances from the large hypothesis space.
  - Specialization - General to Specific learning
  - Generalization - Specific to General learning



Example 3.2: Consider the training instances shown in Table 3.1 and illustrate Specific to General Learning.

Solution: We will start from all false or the most specific hypothesis to determine the most restrictive specialization. Consider only the positive instances and generalize the most specific hypothesis. Ignore the negative instances.

This learning is illustrated as follows:

The most specific hypothesis is taken now, which will not classify any instance to true.

$h =$ <φ φ φ φ φ φ φ φ>

Read the first instance *I*1, to generalize the hypothesis *h* so that this positive instance can be classified by the hypothesis *h*1.

*I*1:	No	Short	Yes	No	No	Black	No	Big Yes (Positive instance)
*h*1 =	<No	Short	Yes	No	No	Black	No	Big>

Example cont..

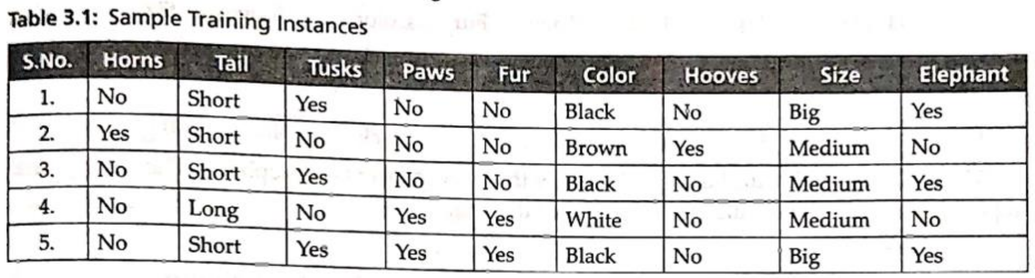
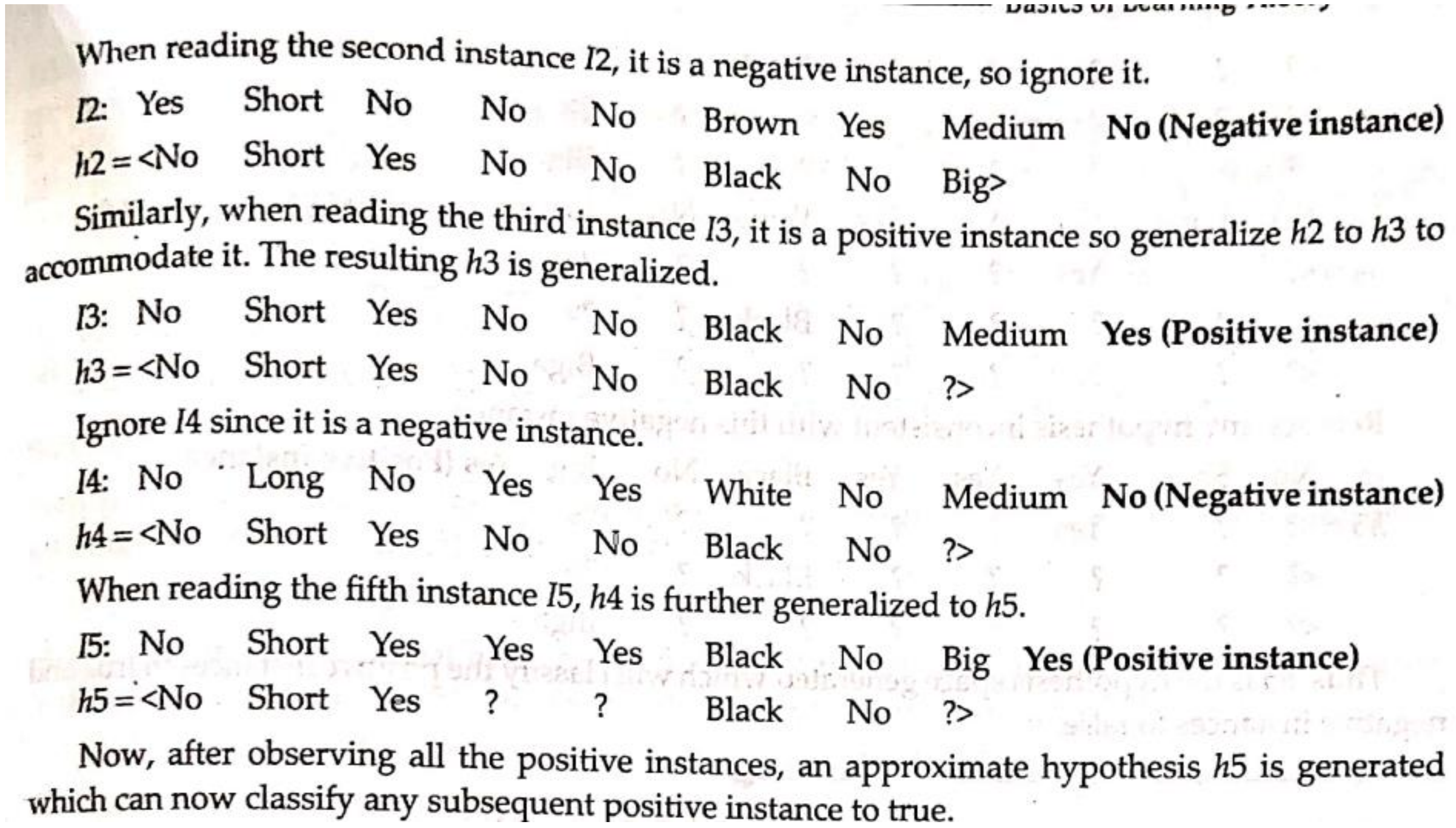


Table 3.1: Sample Training Instances

S.No.	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size	Elephant
1.	No	Short	Yes	No	No	Black	No	Big	Yes
2.	Yes	Short	No	No	No	Brown	Yes	Medium	No
3.	No	Short	Yes	No	No	Black	No	Medium	Yes
4.	No	Long	No	Yes	Yes	White	No	Medium	No
5.	No	Short	Yes	Yes	Yes	Black	No	Big	Yes



When reading the second instance *I*2, it is a negative instance, so ignore it.

*I*2: Yes Short No No No Brown Yes Medium **No (Negative instance)**

*h*2 = <No Short Yes No No Black No Big>

Similarly, when reading the third instance *I*3, it is a positive instance so generalize *h*2 to *h*3 to accommodate it. The resulting *h*3 is generalized.

*I*3: No Short Yes No No Black No Medium **Yes (Positive instance)**

*h*3 = <No Short Yes No No Black No ?>

Ignore *I*4 since it is a negative instance.

*I*4: No Long No Yes Yes White No Medium **No (Negative instance)**

*h*4 = <No Short Yes No No Black No ?>

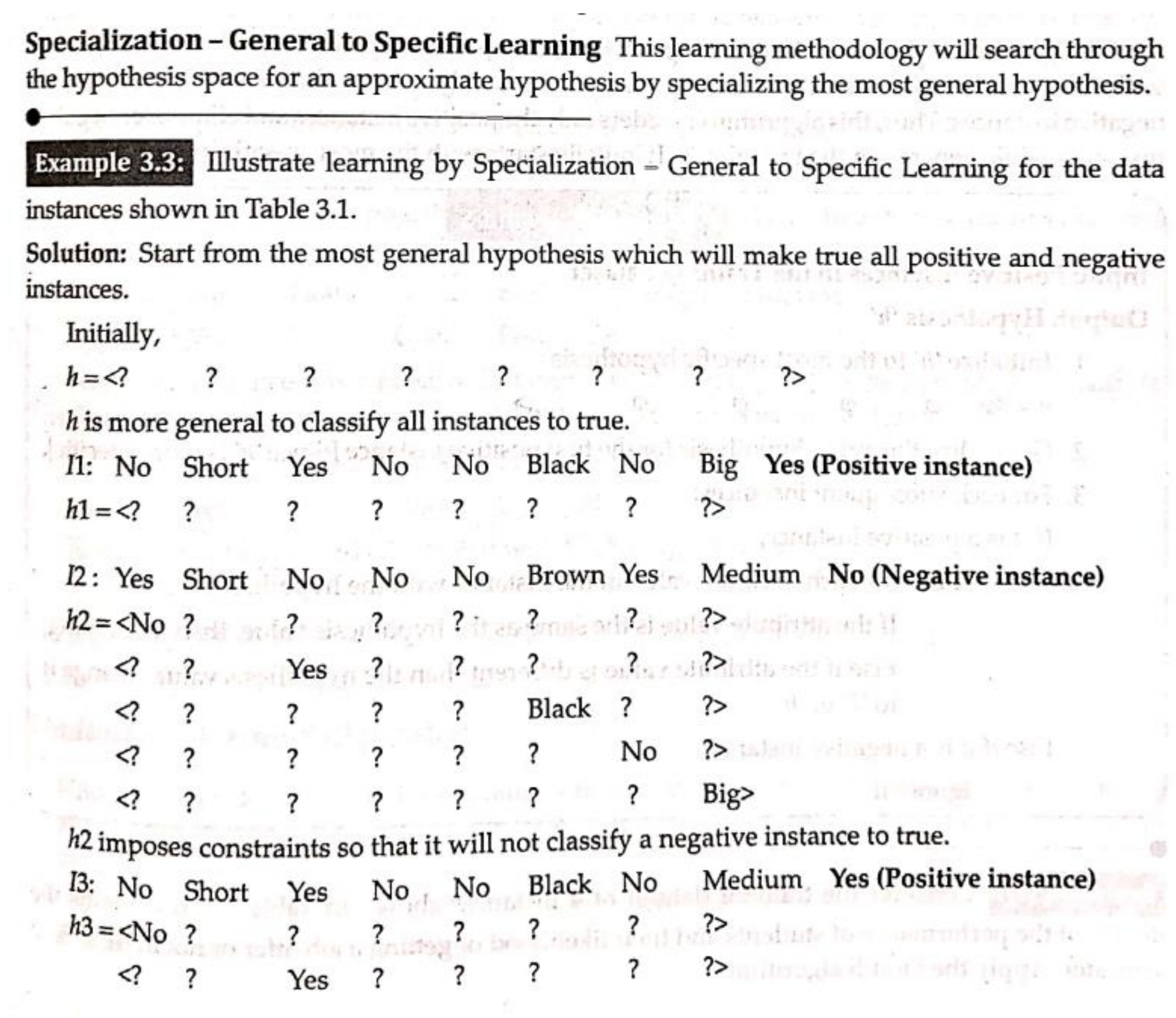
When reading the fifth instance *I*5, *h*4 is further generalized to *h*5.

*I*5: No Short Yes Yes Yes Black No Big **Yes (Positive instance)**

*h*5 = <No Short Yes ? ? Black No ?>

Now, after observing all the positive instances, an approximate hypothesis *h*5 is generated which can now classify any subsequent positive instance to true.

Specialization General to specific Learning



Specialization – General to Specific Learning This learning methodology will search through the hypothesis space for an approximate hypothesis by specializing the most general hypothesis.

Example 3.3: Illustrate learning by Specialization – General to Specific Learning for the data instances shown in Table 3.1.

Solution: Start from the most general hypothesis which will make true all positive and negative instances.

Initially,

h = <? ? ? ? ? ? ? ?>

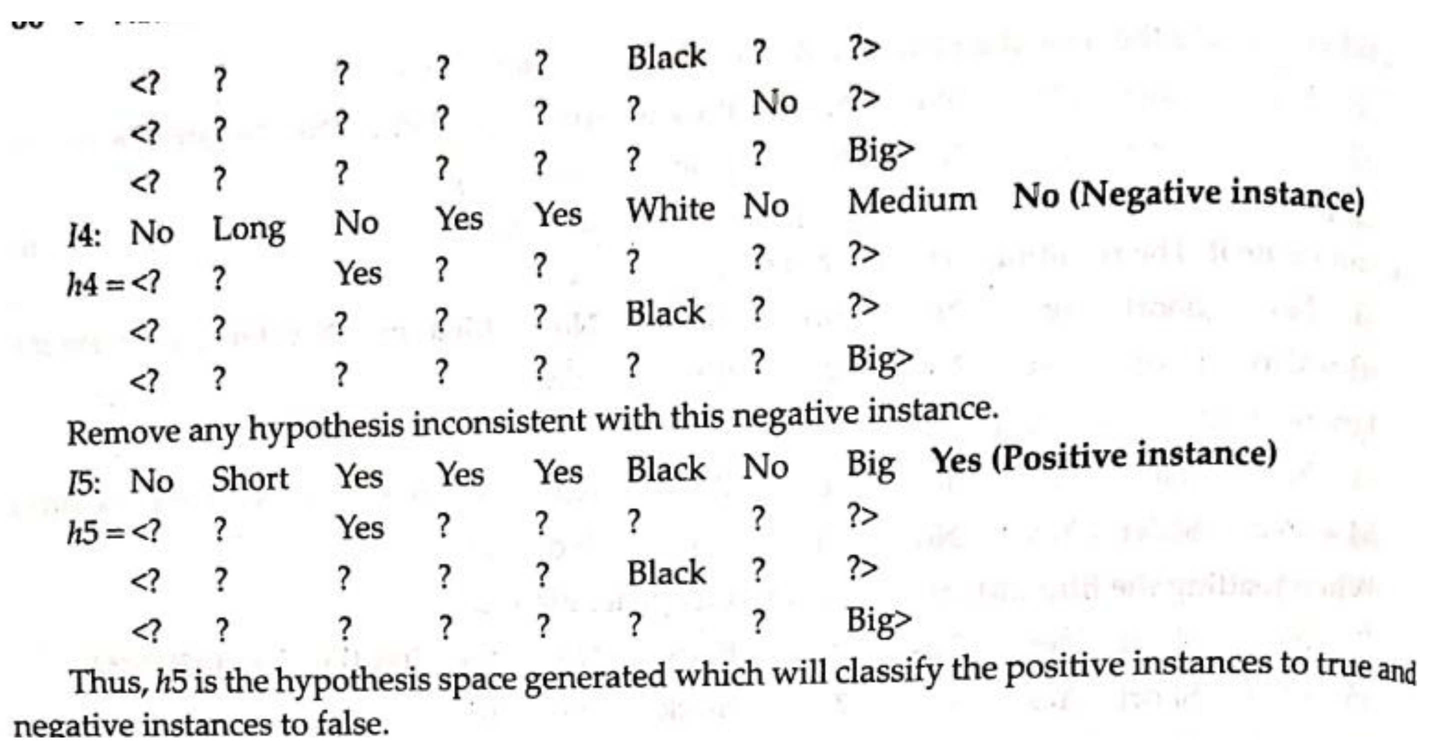
h is more general to classify all instances to true.

*I*1:	No	Short	Yes	No	No	Black	No	Big	**Yes (Positive instance)**
*h*1 =	<?	?	?	?	?	?	?	?>	
*I*2:	Yes	Short	No	No	No	Brown	Yes	Medium	**No (Negative instance)**
*h*2 =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	No	?>	
	<?	?	?	?	?	?	?	Big>	

*h*2 imposes constraints so that it will not classify a negative instance to true.

*I*3:	No	Short	Yes	No	No	Black	No	Medium	**Yes (Positive instance)**
*h*3 =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	

Specialization General to specific Learning



	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	No	?>	
	<?	?	?	?	?	?	?	Big>	
*I*4:	No	Long	No	Yes	Yes	White	No	Medium	**No (Negative instance)**
*h*4 =	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	?	Big>	

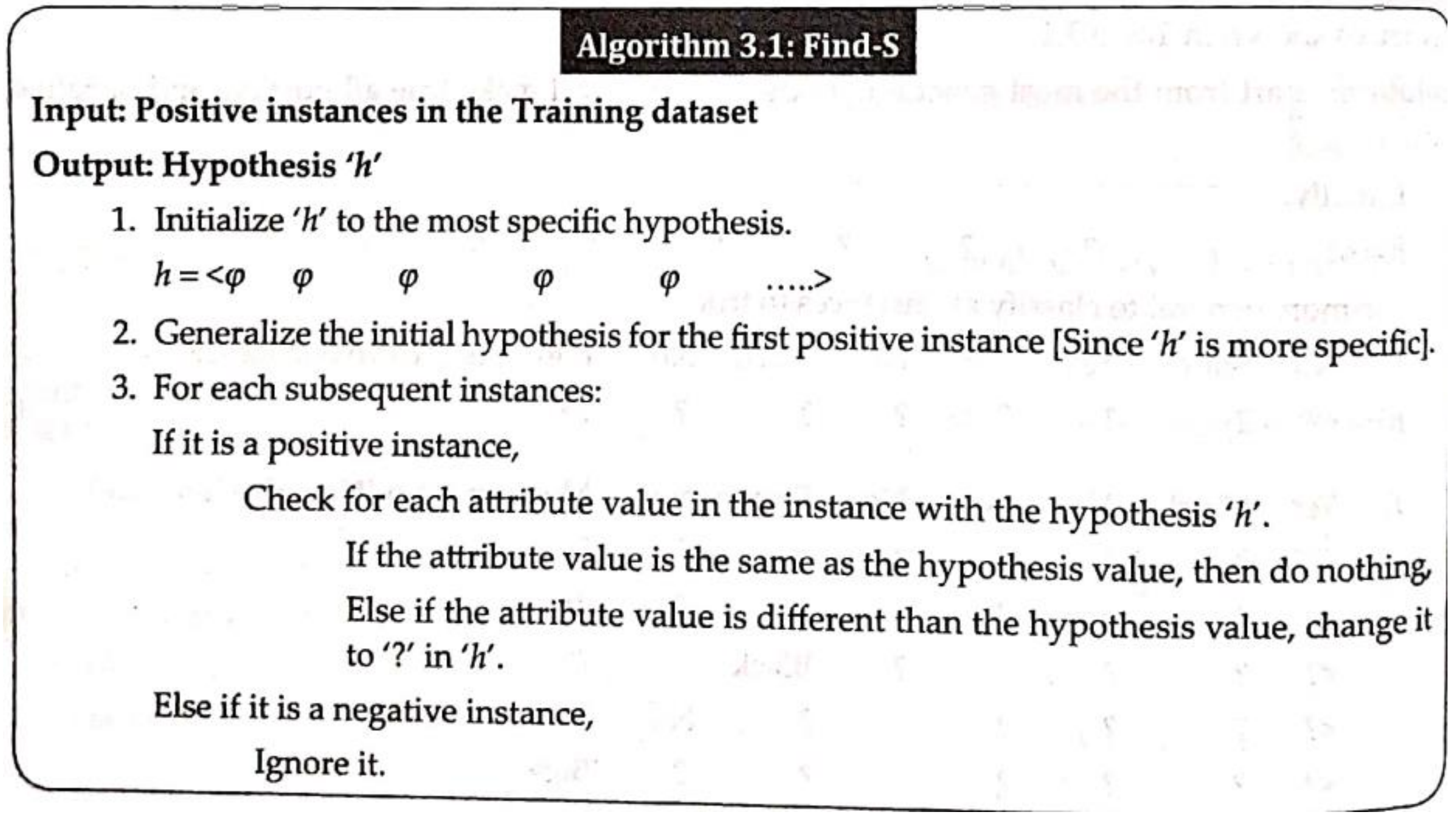
Remove any hypothesis inconsistent with this negative instance.

*I*5:	No	Short	Yes	Yes	Yes	Black	No	Big	**Yes (Positive instance)**
*h*5 =	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	?	Big>	

Thus, *h*5 is the hypothesis space generated which will classify the positive instances to true and negative instances to false.

Hypothesis Space Search by Find-S Algorithm

- Find-S algorithm is guaranteed to converge to the most specific hypothesis in H that is consistent with the positive instances in the training dataset.
- Obviously, it will also be consistent with the negative instances. Thus, this algorithm considers only the positive instances and eliminates negative instances while generating the hypothesis. It initially starts with the most specific hypothesis.



Algorithm 3.1: Find-S

Input: Positive instances in the Training dataset

Output: Hypothesis '*h*'

```
1. Initialize 'h' to the most specific hypothesis.
   $h = <\varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad .....>$
2. Generalize the initial hypothesis for the first positive instance [Since 'h' is more specific].
3. For each subsequent instances:
   If it is a positive instance,
       Check for each attribute value in the instance with the hypothesis 'h'.
           If the attribute value is the same as the hypothesis value, then do nothing.
           Else if the attribute value is different than the hypothesis value, change it
           to '?' in 'h'.
   Else if it is a negative instance,
       Ignore it.
```

Example

- Consider the training dataset of 4 instances shown in Table 3.2. It contains the details of the performance of students and their likelihood of getting a job offer or not in their final semester. Apply the Find-S algorithm.

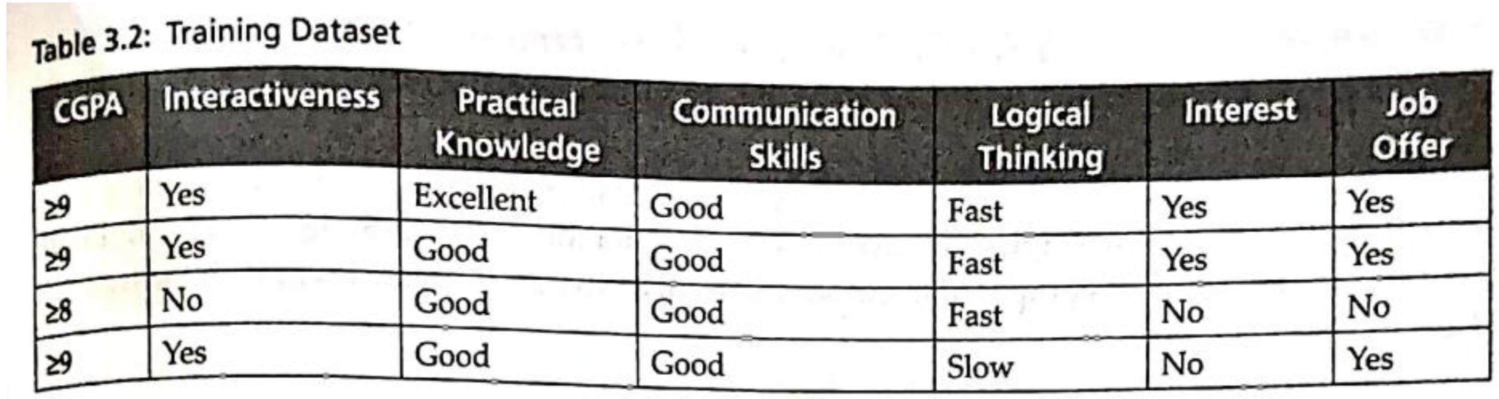
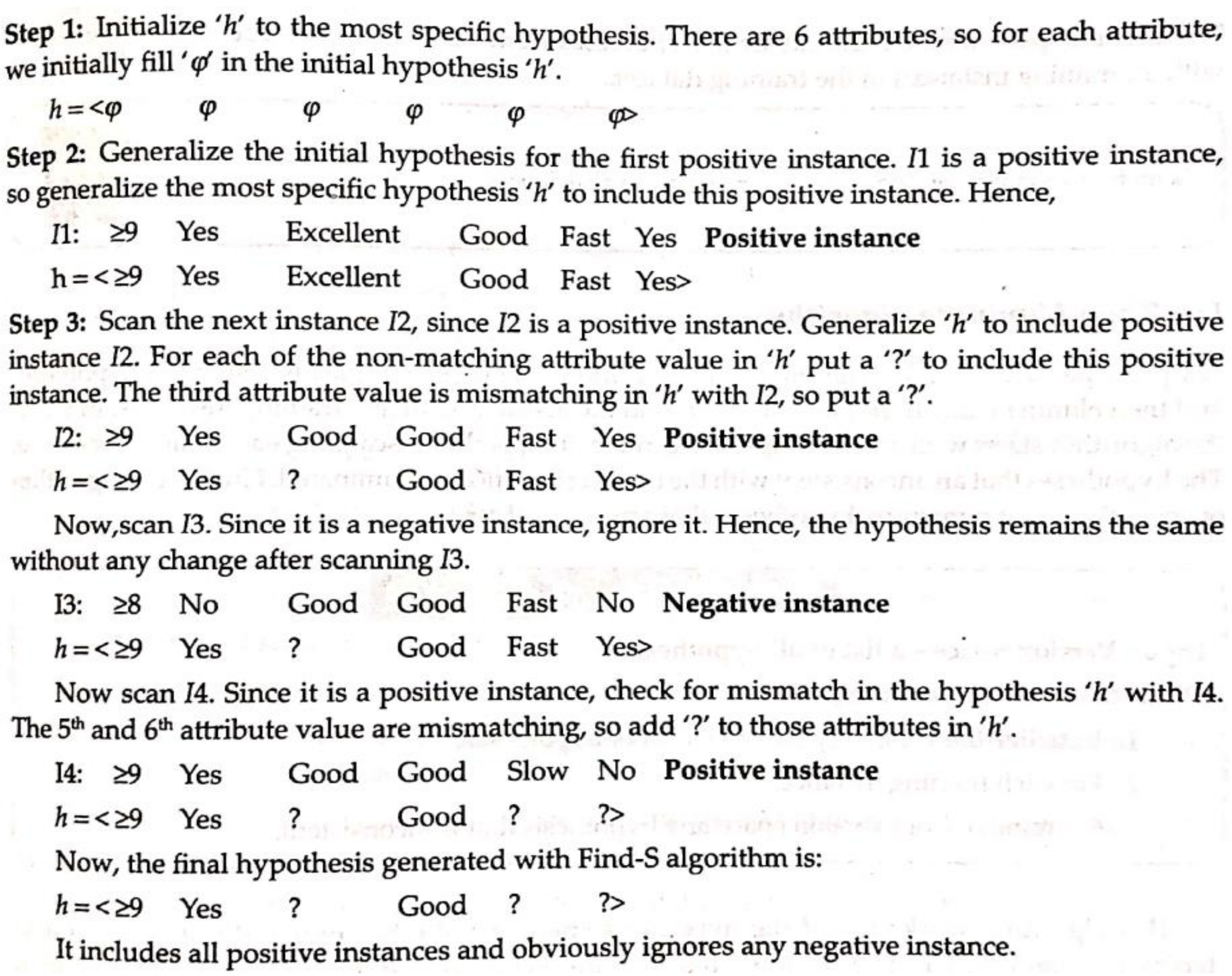


Table 3.2: Training Dataset

CGPA	Interactiveness	Practical Knowledge	Communication Skills	Logical Thinking	Interest	Job Offer
≥9	Yes	Excellent	Good	Fast	Yes	Yes
≥9	Yes	Good	Good	Fast	Yes	Yes
≥8	No	Good	Good	Fast	No	No
≥9	Yes	Good	Good	Slow	No	Yes

Solution



Step 1: Initialize 'h' to the most specific hypothesis. There are 6 attributes, so for each attribute, we initially fill 'φ' in the initial hypothesis 'h'.

$h = <\varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi>$

Step 2: Generalize the initial hypothesis for the first positive instance. $I1$ is a positive instance, so generalize the most specific hypothesis 'h' to include this positive instance. Hence,

$I1$:	≥ 9	Yes	Excellent	Good	Fast	Yes	**Positive instance**
$h = <$	≥ 9	Yes	Excellent	Good	Fast	Yes>	

Step 3: Scan the next instance $I2$, since $I2$ is a positive instance. Generalize 'h' to include positive instance $I2$. For each of the non-matching attribute value in 'h' put a '?' to include this positive instance. The third attribute value is mismatching in 'h' with $I2$, so put a '?'.

$I2$:	≥ 9	Yes	Good	Good	Fast	Yes	**Positive instance**
$h = <$	≥ 9	Yes	?	Good	Fast	Yes>	

Now,scan $I3$. Since it is a negative instance, ignore it. Hence, the hypothesis remains the same without any change after scanning $I3$.

I3:	≥ 8	No	Good	Good	Fast	No	**Negative instance**
$h = <$	≥ 9	Yes	?	Good	Fast	Yes>	

Now scan $I4$. Since it is a positive instance, check for mismatch in the hypothesis 'h' with $I4$. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in 'h'.

I4:	≥ 9	Yes	Good	Good	Slow	No	**Positive instance**
$h = <$	≥ 9	Yes	?	Good	?	?>	

Now, the final hypothesis generated with Find-S algorithm is:

$h = < \geq 9 \quad \text{Yes} \quad ? \quad \text{Good} \quad ? \quad ?>$

It includes all positive instances and obviously ignores any negative instance.

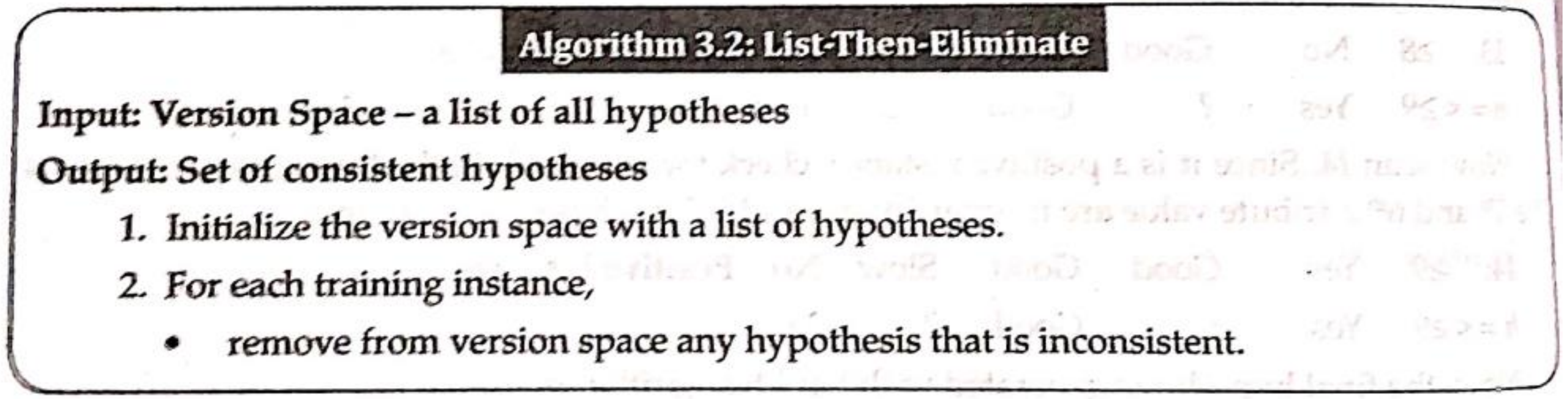
Limitations of Find-S Algorithm

- Find-S algorithm tries to find a hypothesis that is consistent with positive instances, ignoring all negative instances. As long as the training dataset is consistent, the hypothesis found by this algorithm may be consistent.
- The algorithm finds only one unique hypothesis, wherein there may be many other hypotheses that are consistent with the training dataset.
- Many times, the training dataset may contain some errors; hence such inconsistent data instane can mislead this algorithm in determining the consistent hypothesis since it ignores negative instances.

Version Spaces

- The version space contains the subset of hypotheses from the hypothesis space that is consistent with all training instances in the training dataset.

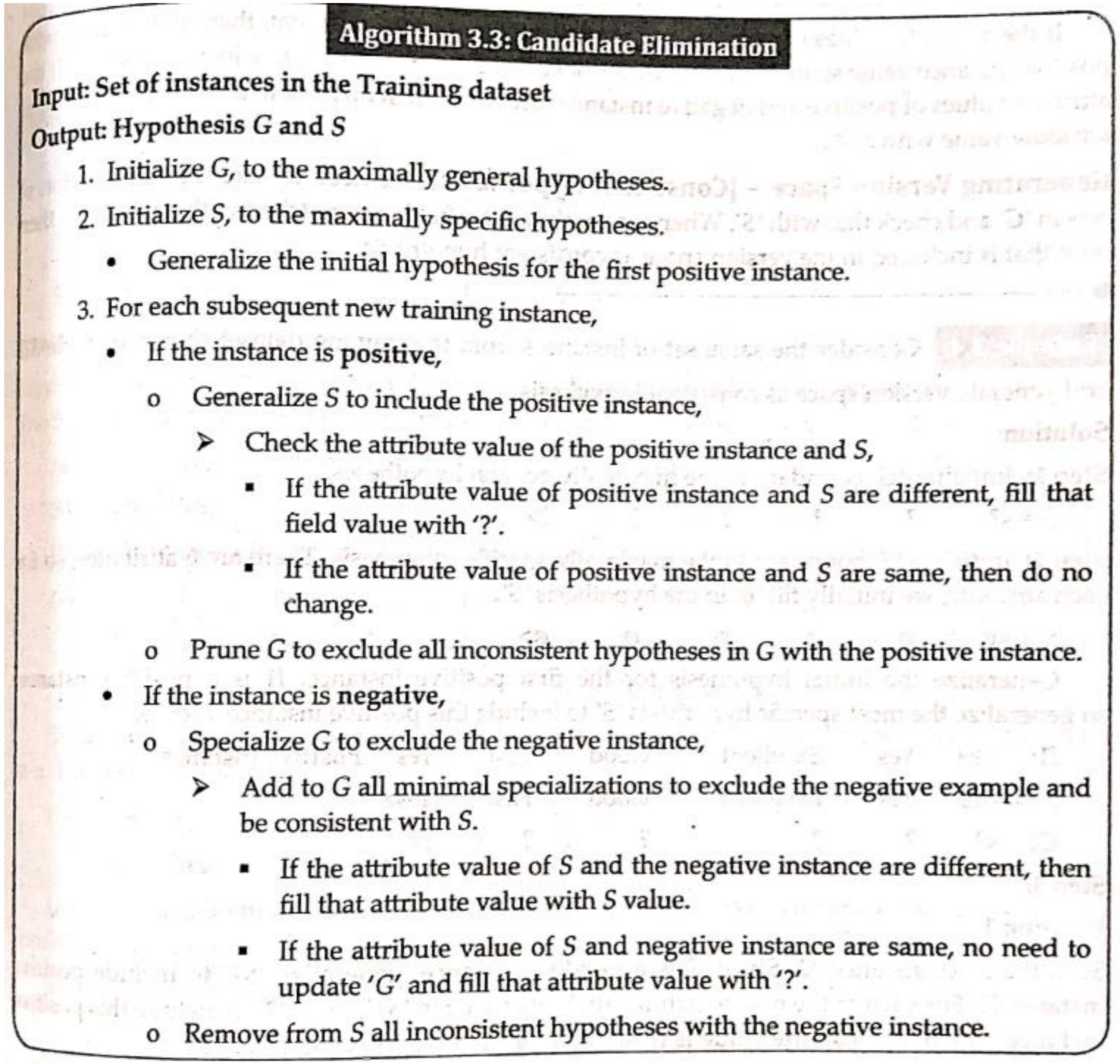
- List-Then-Eliminate Algorithm
  - The principle idea of this learning algorithm is to initialize the version space to contain all hypotheses and then eliminate any hypothesis that is found inconsistent with any training instances. Initially, the algorithm starts with a version space to contain all hypotheses scanning each training instance. The hypotheses that are inconsistent with the training instance are eliminated. Finally, the algorithm outputs the list of remaining hypotheses that are all consistent.



Algorithm 3.2: List-Then-Eliminate

```
Input: Version Space – a list of all hypotheses
Output: Set of consistent hypotheses
    1. Initialize the version space with a list of hypotheses.
    2. For each training instance,
        • remove from version space any hypothesis that is inconsistent.
```

Algorithm



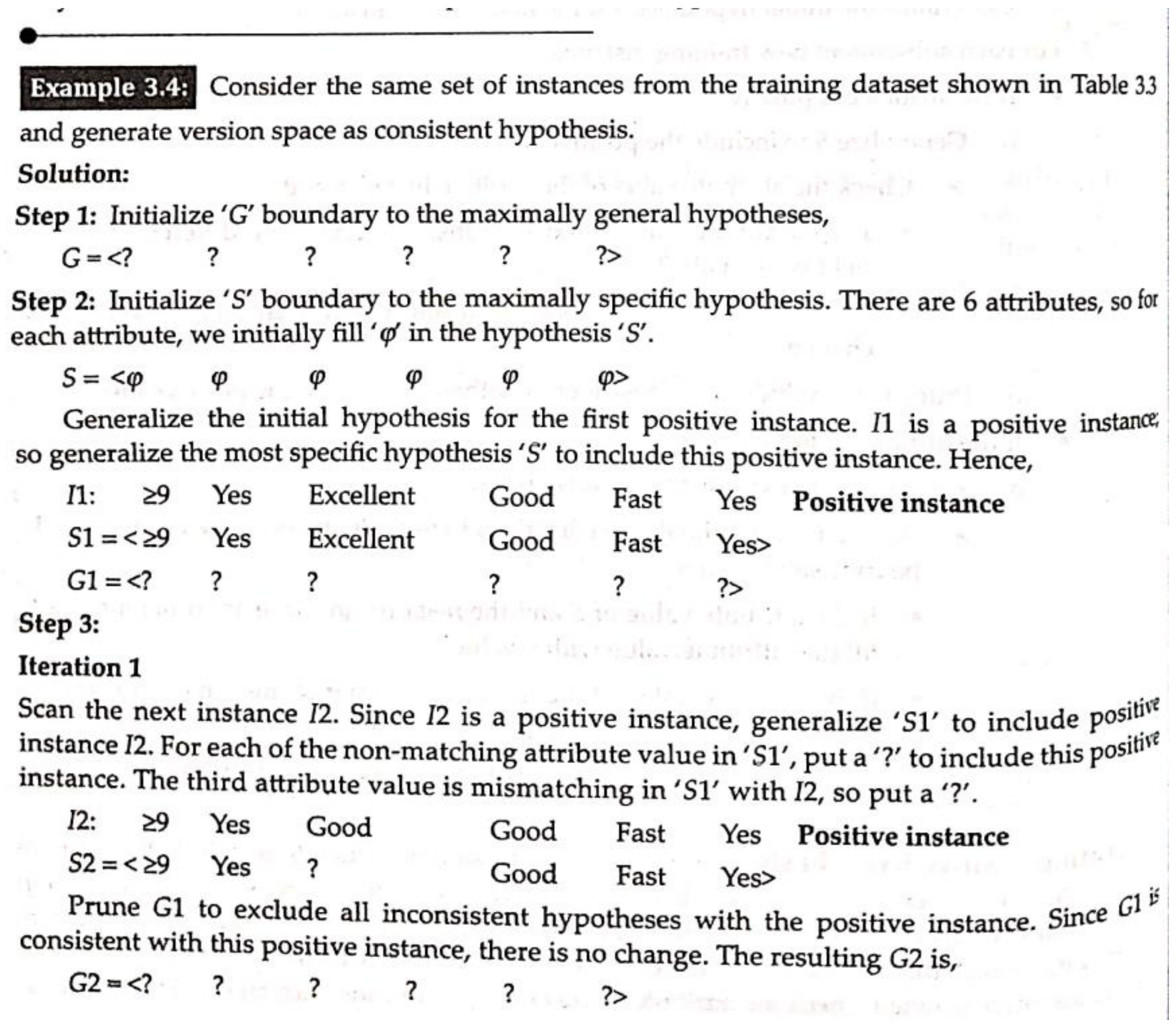
Algorithm 3.3: Candidate Elimination

Input: Set of instances in the Training dataset

Output: Hypothesis *G* and *S*

1. Initialize *G*, to the maximally general hypotheses.
2. Initialize *S*, to the maximally specific hypotheses.
   - Generalize the initial hypothesis for the first positive instance.
3. For each subsequent new training instance,
   - If the instance is **positive,**
     - o Generalize *S* to include the positive instance,
       - ➢ Check the attribute value of the positive instance and *S*,
         - ▪ If the attribute value of positive instance and *S* are different, fill that field value with '?'.
         - ▪ If the attribute value of positive instance and *S* are same, then do no change.
     - o Prune *G* to exclude all inconsistent hypotheses in *G* with the positive instance.
   - If the instance is **negative,**
     - o Specialize *G* to exclude the negative instance,
       - ➢ Add to *G* all minimal specializations to exclude the negative example and be consistent with *S*.
         - ▪ If the attribute value of *S* and the negative instance are different, then fill that attribute value with *S* value.
         - ▪ If the attribute value of *S* and negative instance are same, no need to update '*G*' and fill that attribute value with '?'.
     - o Remove from *S* all inconsistent hypotheses with the negative instance.

Example



Example 3.4: Consider the same set of instances from the training dataset shown in Table 3.3 and generate version space as consistent hypothesis.

Solution:

Step 1: Initialize '*G*' boundary to the maximally general hypotheses,

G = <? ? ? ? ? ?>

Step 2: Initialize '*S*' boundary to the maximally specific hypothesis. There are 6 attributes, so for each attribute, we initially fill '*φ*' in the hypothesis '*S*'.

S = <φ φ φ φ φ φ>

Generalize the initial hypothesis for the first positive instance. *I*1 is a positive instance; so generalize the most specific hypothesis '*S*' to include this positive instance. Hence,

*I*1: ≥9 Yes Excellent Good Fast Yes **Positive instance**

*S*1 = <≥9 Yes Excellent Good Fast Yes>

*G*1 = <? ? ? ? ? ?>

Step 3:

Iteration 1

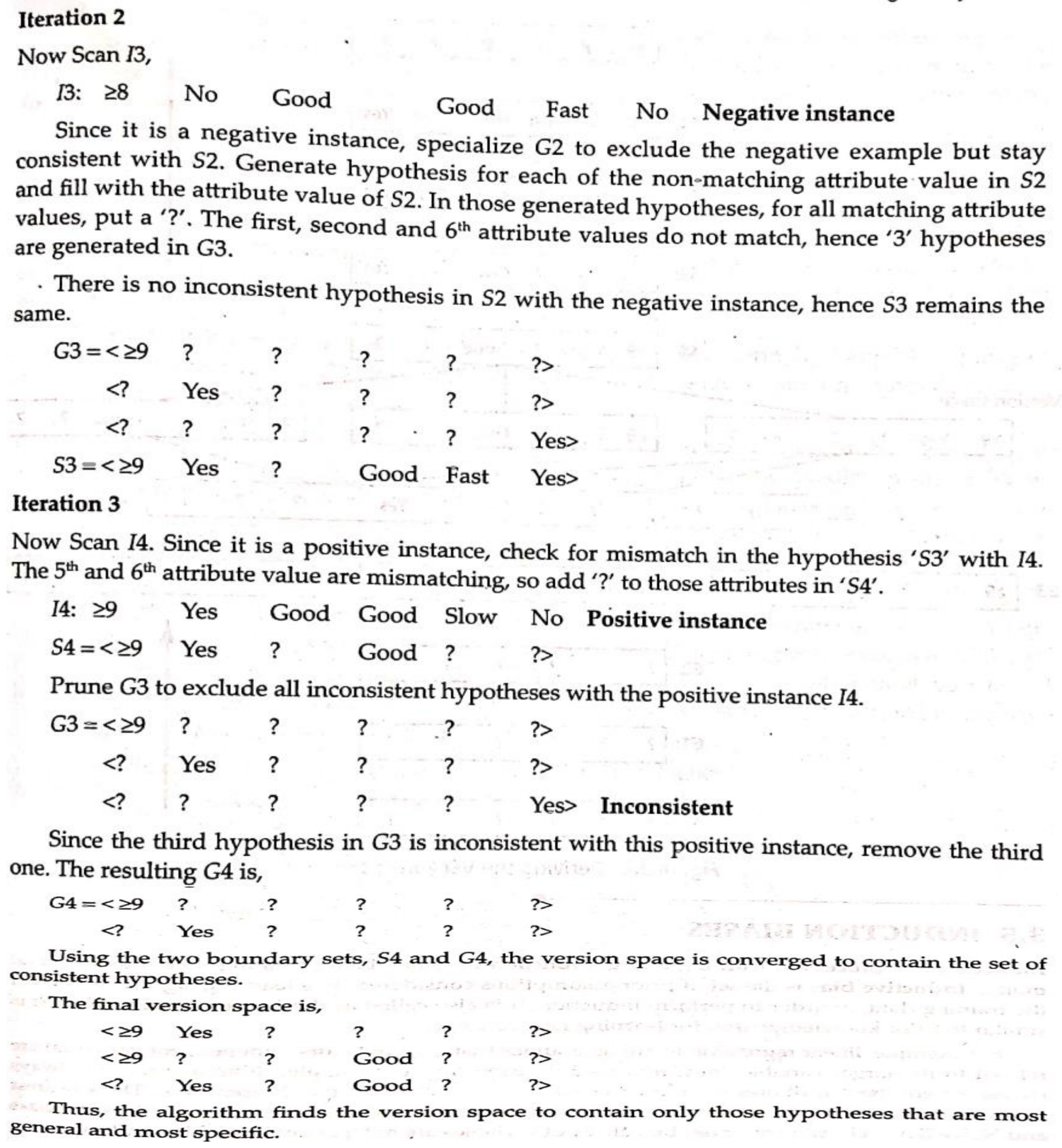
Scan the next instance *I*2. Since *I*2 is a positive instance, generalize '*S*1' to include positive instance *I*2. For each of the non-matching attribute value in '*S*1', put a '?' to include this positive instance. The third attribute value is mismatching in '*S*1' with *I*2, so put a '?'.

*I*2: ≥9 Yes Good Good Fast Yes **Positive instance**

*S*2 = <≥9 Yes ? Good Fast Yes>

Prune *G*1 to exclude all inconsistent hypotheses with the positive instance. Since *G*1 is consistent with this positive instance, there is no change. The resulting *G*2 is,

*G*2 = <? ? ? ? ? ?>



Iteration 2

Now Scan *I*3,

*I*3: ≥8 No Good Good Fast No **Negative instance**

Since it is a negative instance, specialize *G*2 to exclude the negative example but stay consistent with *S*2. Generate hypothesis for each of the non-matching attribute value in *S*2 and fill with the attribute value of *S*2. In those generated hypotheses, for all matching attribute values, put a '?'. The first, second and 6th attribute values do not match, hence '3' hypotheses are generated in *G*3.

There is no inconsistent hypothesis in *S*2 with the negative instance, hence *S*3 remains the same.

*G*3 = <≥9 ? ? ? ? ?>
<? Yes ? ? ? ?>
<? ? ? ? ? Yes>

*S*3 = <≥9 Yes ? Good Fast Yes>

Iteration 3

Now Scan *I*4. Since it is a positive instance, check for mismatch in the hypothesis '*S*3' with *I*4. The 5th and 6th attribute value are mismatching, so add '?' to those attributes in '*S*4'.

*I*4: ≥9 Yes Good Good Slow No **Positive instance**

*S*4 = <≥9 Yes ? Good ? ?>

Prune *G*3 to exclude all inconsistent hypotheses with the positive instance *I*4.

*G*3 = <≥9 ? ? ? ? ?>
<? Yes ? ? ? ?>
<? ? ? ? ? Yes> **Inconsistent**

Since the third hypothesis in *G*3 is inconsistent with this positive instance, remove the third one. The resulting *G*4 is,

*G*4 = <≥9 ? ? ? ? ?>
<? Yes ? ? ? ?>

Using the two boundary sets, *S*4 and *G*4, the version space is converged to contain the set of consistent hypotheses.

The final version space is,

<≥9 Yes ? ? ? ?>
<≥9 ? ? Good ? ?>
<? Yes ? Good ? ?>

Thus, the algorithm finds the version space to contain only those hypotheses that are most general and most specific.

Similarity Based Learning

Module 3

Chapter 2

INTRODUCTION TO SIMILARITY OR INSTANCE-BASEDLEARNING

- Similarity-based classifiers use similarity measures to locate the nearest neighbors and classify a test instance which works in contrast with other learning mechanisms such as decision trees or neural networks.
- Similarity-based learning is also called as Instance-based learning/Just-in time learning since it does not build an abstract model of the training instances and performs lazy learning when classifying a new instance.
- This learning mechanism simply stores all data and uses it only when it needs to classify an unseen instance. The advantage of using this learning is that processing occurs only when a request to classify a new instance is given.
- This methodology is particularly useful when the whole dataset is not available in the beginning but collected in an incremental manner.

INTRODUCTION TO SIMILARITY OR INSTANCE-BASEDLEARNING

- The drawback of this learning is that it requires a large memory to store the data since a global abstract model is not constructed initially with the training data.
- Classification of instances is done based on the measure of similarity in the form of distance functions over data instances.
- Several distance metrics are used to estimate the similarity or dissimilarity between instances required for clustering, nearest neighbor classification, anomaly detection, and so on.
- Popular distance metrics used are Hamming distance, Euclidean distance, Manhattan distance, Minkowski distance, Cosine similarity, Mahalanobis distance, Pearson's correlation or correlation similarity, Mean squared difference, Jaccard coefficient, Tanimoto coefficient, etc.

Difference Between

Instance-based Learning	Model-based Learning
Lazy Learners	Eager Learners
Processing of training instances is done only during testing phase	Processing of training instances is done during training phase
No model is built with the training instances before it receives a test instance	Generalizes a model with the training instances before it receives a test instance
Predicts the class of the test instance directly from the training data	Predicts the class of the test instance from the model built
Slow in testing phase	Fast in testing phase
Learns by making many local approximations	Learns by creating global approximation

Instance Based Learning

- Instance-based learning also comes under the category of memory-based models which normally compare the given test instance with the trained instances that are stored in memory.
- Memory-based models classify a test instance by checking the similarity with the training instances.
- Some instance based algorithm
    - k-Nearest Neighbor (k-NN)
    - Variants of Nearest Neighbor learning
    - Locally Weighted Regression
    - Learning Vector Quantization (LVQ)
    - Self-Organizing Map (SOM)
    - Radial Basis Function (RBF) networks

NEAREST-NEIGHBOR LEARNING

- A natural approach to similarity-based classification is k-Nearest-Neighbors (k-NN), which is a non-parametric method used for both classification and regression problems.
- It is a simple and powerful non-parametric algorithm that predicts the category of the test instance according to the 'k' training samples which are closer to the test instance and classifies it to that category which has the largest probability.
- A visual representation of this learning is shown in Figure 4.1.
- There are two classes of objects called C1 and C2 in the given figure.
- When given a test instance T, the category of this test instance is determined by looking at the class of k = 3 nearest neighbors. Thus, the class of this test instance T is predicted as C2.

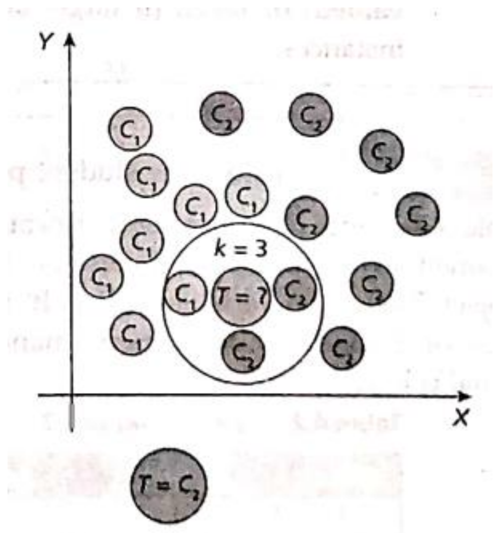


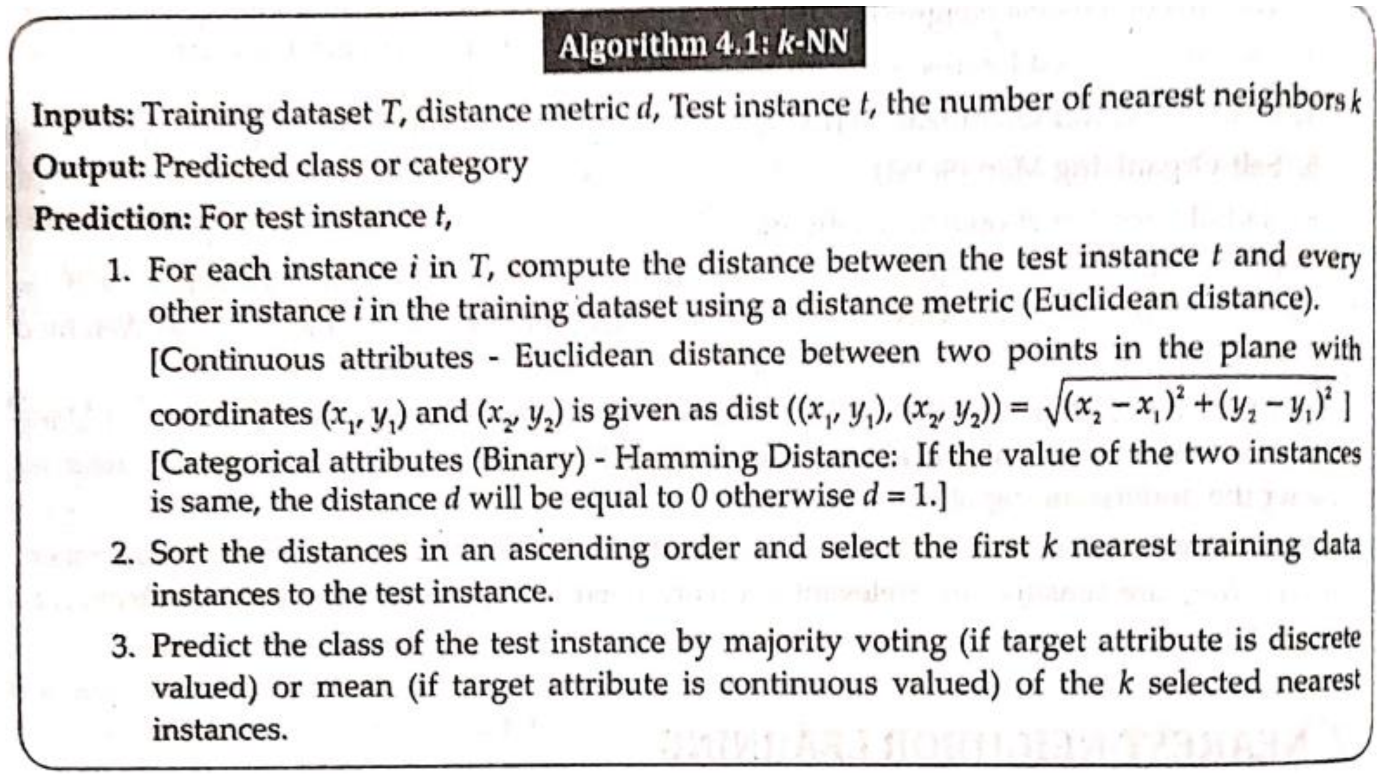
Figure 4.1: Visual Representation of *k*-Nearest Neighbor Learning

NEAREST-NEIGHBOR LEARNING

- The algorithm relies on the assumption that similar objects are close to each other in the feature space.
- k-NN performs instance-based learning which just stores the training data instances and learning instances case by case.
- The model is also 'memory-based' as it uses training data at time when predictions need to be made. It is a lazy learning algorithm since no prediction model is built earlier with training instances and classification happens only after getting the test instance.
- The algorithm classifies a new instance by determining the 'k' most similar instances (i.e., k nearest neighbors) and summarizing the output of those 'k' instances.
- If the target variable is discrete then it is a classification problem, so it selects the most common class value among the ‘k’ instances by a majority vote.
- However, if the target variable is continuous then it is a regression problem, and hence the mean output variable of the 'k' instances is the output of the test instance.

NEAREST-NEIGHBOR LEARNING

- The most popular distance measure such as Euclidean distance is used in k-NN to determine the 'k' instances which are similar to the test instance. The value of 'k' is best determined by tuning with different 'k' values and choosing the 'k' which classifies the test instance more accurately.



Algorithm 4.1: *k*-NN

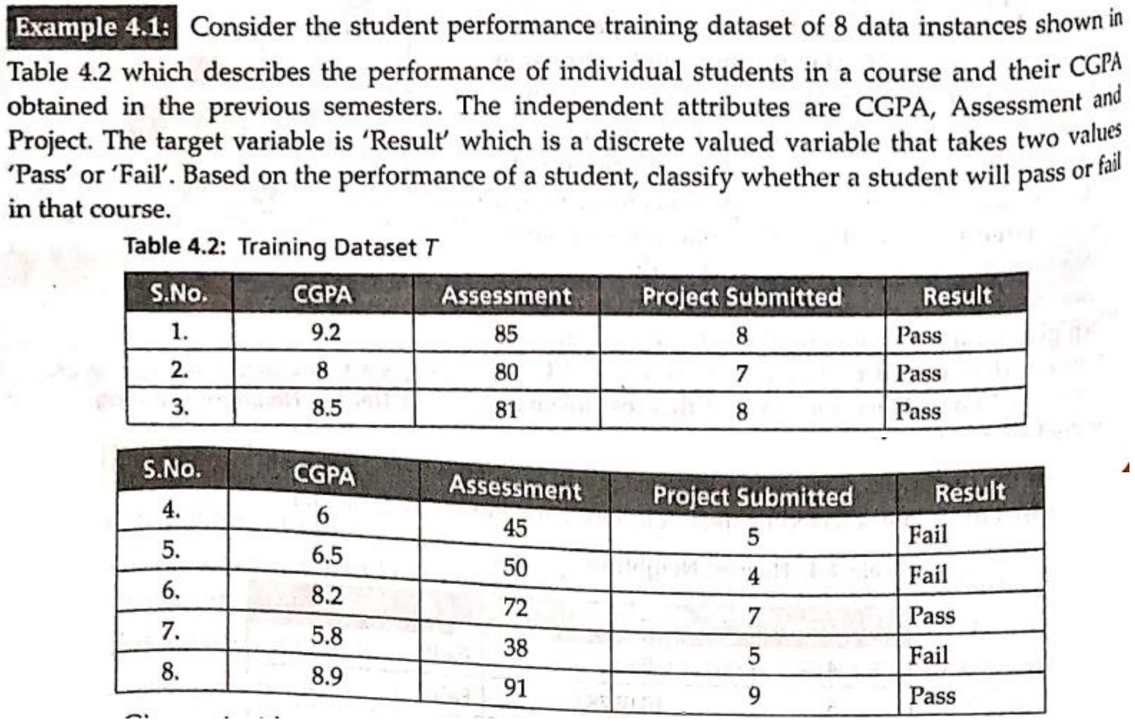
Inputs: Training dataset T, distance metric d, Test instance t, the number of nearest neighbors k

Output: Predicted class or category

Prediction: For test instance t,

1. For each instance i in T, compute the distance between the test instance t and every other instance i in the training dataset using a distance metric (Euclidean distance).
   [Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as dist $((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]
   [Categorical attributes (Binary) - Hamming Distance: If the value of the two instances is same, the distance d will be equal to 0 otherwise $d = 1$.]
2. Sort the distances in an ascending order and select the first k nearest training data instances to the test instance.
3. Predict the class of the test instance by majority voting (if target attribute is discrete valued) or mean (if target attribute is continuous valued) of the k selected nearest instances.

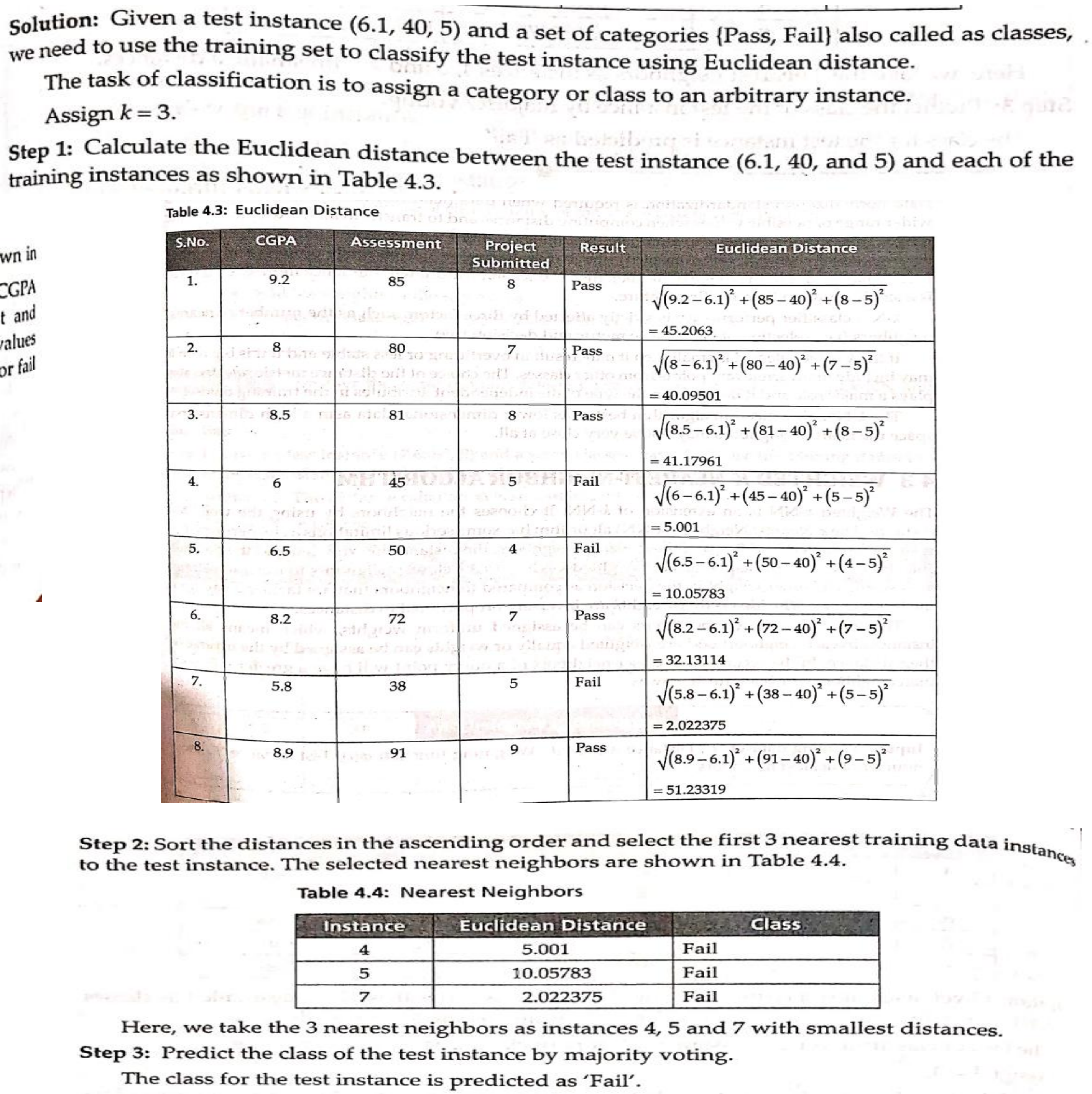
Example



Example 4.1: Consider the student performance training dataset of 8 data instances shown in Table 4.2 which describes the performance of individual students in a course and their CGPA obtained in the previous semesters. The independent attributes are CGPA, Assessment and Project. The target variable is 'Result' which is a discrete valued variable that takes two values 'Pass' or 'Fail'. Based on the performance of a student, classify whether a student will pass or fail in that course.

Table 4.2: Training Dataset *T*

S.No.	CGPA	Assessment	Project Submitted	Result
1.	9.2	85	8	Pass
2.	8	80	7	Pass
3.	8.5	81	8	Pass
4.	6	45	5	Fail
5.	6.5	50	4	Fail
6.	8.2	72	7	Pass
7.	5.8	38	5	Fail
8.	8.9	91	9	Pass



Solution: Given a test instance (6.1, 40, 5) and a set of categories {Pass, Fail} also called as classes, we need to use the training set to classify the test instance using Euclidean distance.

The task of classification is to assign a category or class to an arbitrary instance.

Assign $k = 3$.

Step 1: Calculate the Euclidean distance between the test instance (6.1, 40, and 5) and each of the training instances as shown in Table 4.3.

Table 4.3: Euclidean Distance

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-6.1)^2+(85-40)^2+(8-5)^2}$ $= 45.2063$
2.	8	80	7	Pass	$\sqrt{(8-6.1)^2+(80-40)^2+(7-5)^2}$ $= 40.09501$
3.	8.5	81	8	Pass	$\sqrt{(8.5-6.1)^2+(81-40)^2+(8-5)^2}$ $= 41.17961$
4.	6	45	5	Fail	$\sqrt{(6-6.1)^2+(45-40)^2+(5-5)^2}$ $= 5.001$
5.	6.5	50	4	Fail	$\sqrt{(6.5-6.1)^2+(50-40)^2+(4-5)^2}$ $= 10.05783$
6.	8.2	72	7	Pass	$\sqrt{(8.2-6.1)^2+(72-40)^2+(7-5)^2}$ $= 32.13114$
7.	5.8	38	5	Fail	$\sqrt{(5.8-6.1)^2+(38-40)^2+(5-5)^2}$ $= 2.022375$
8.	8.9	91	9	Pass	$\sqrt{(8.9-6.1)^2+(91-40)^2+(9-5)^2}$ $= 51.23319$

Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.4.

Table 4.4: Nearest Neighbors

Instance	Euclidean Distance	Class
4	5.001	Fail
5	10.05783	Fail
7	2.022375	Fail

Here, we take the 3 nearest neighbors as instances 4, 5 and 7 with smallest distances.

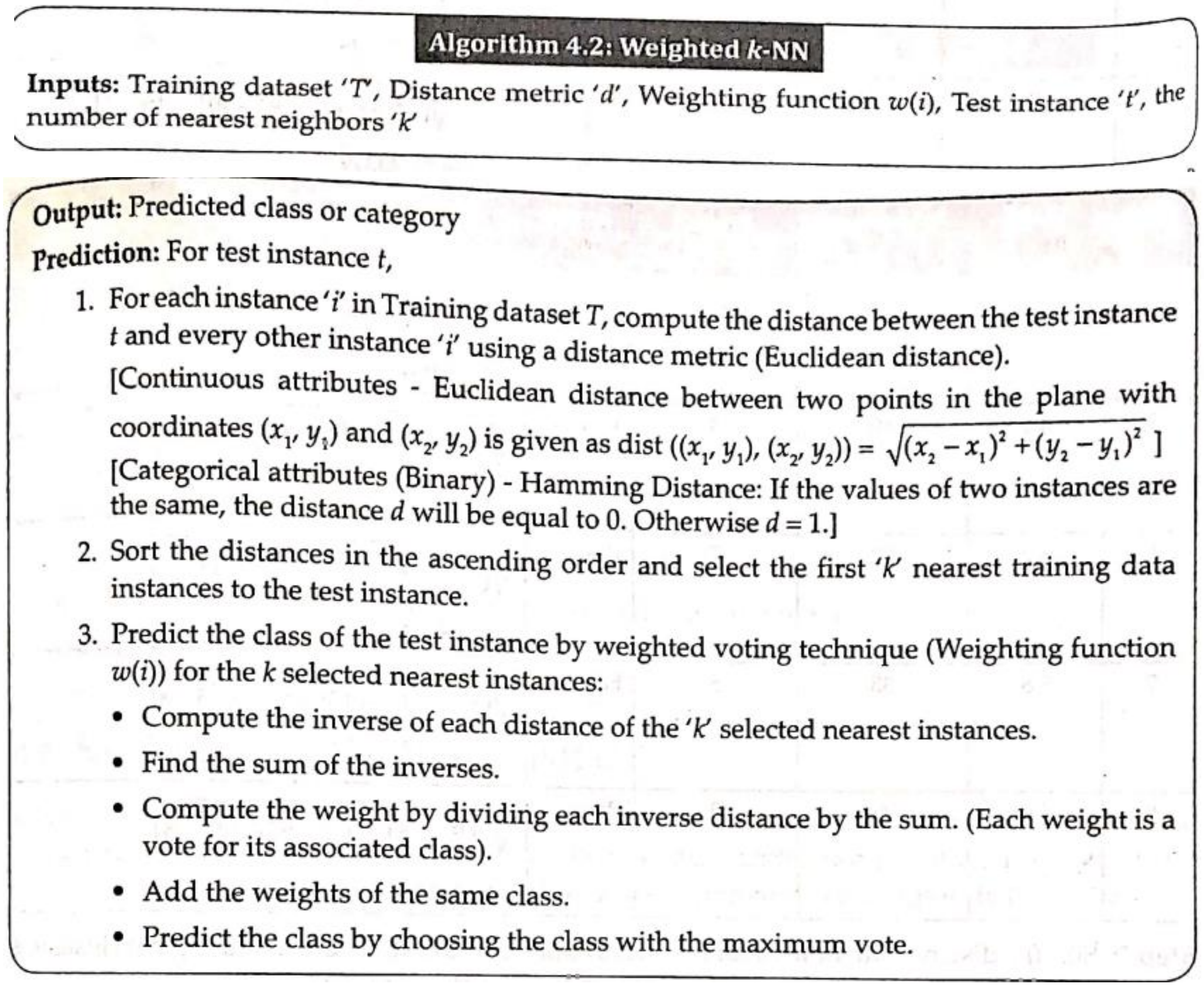
Step 3: Predict the class of the test instance by majority voting.

The class for the test instance is predicted as 'Fail'.

WEIGHTED K-NEAREST-NEIGHBOR ALGORITHM

- The Weighted k-NN is an extension of k-NN. It chooses the neighbors by using the weighted distance.
- The k-Nearest Neighbor (k-NN) algorithm has some serious limitations as its performance is solely dependent on choosing the k nearest neighbors, the distance metric used and the decision rule.
- However, the principle idea of Weighted k-NN is that k closest neighbors to the test instance are assigned a higher weight in the decision as compared to neighbors that are farther away from the test instance. The idea is that weights are inversely proportional to distances.
- The selected k nearest neighbors can be assigned uniform weights, which means all the instances in each neighborhood are weighted equally or weights can be assigned by the inverse of their distance. In the second case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

Algorithm



Algorithm 4.2: Weighted k-NN

Inputs: Training dataset 'T', Distance metric 'd', Weighting function $w(i)$, Test instance 't', the number of nearest neighbors 'k'

Output: Predicted class or category

Prediction: For test instance t,

1. For each instance 'i' in Training dataset T, compute the distance between the test instance t and every other instance 'i' using a distance metric (Euclidean distance).
   [Continuous attributes - Euclidean distance between two points in the plane with coordinates (x_1, y_1) and (x_2, y_2) is given as dist $((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$]
   [Categorical attributes (Binary) - Hamming Distance: If the values of two instances are the same, the distance d will be equal to 0. Otherwise $d = 1$.]
2. Sort the distances in the ascending order and select the first 'k' nearest training data instances to the test instance.
3. Predict the class of the test instance by weighted voting technique (Weighting function $w(i)$) for the k selected nearest instances:
   - Compute the inverse of each distance of the 'k' selected nearest instances.
   - Find the sum of the inverses.
   - Compute the weight by dividing each inverse distance by the sum. (Each weight is a vote for its associated class).
   - Add the weights of the same class.
   - Predict the class by choosing the class with the maximum vote.

Example

- Consider the same training dataset given in Table 4.1. Use Weighted k-NN and determine the class.
- Solution:
  - Step 1: Given a test instance (7.6, 60, 8) and a set of classes {Pass, Fail}, use the training dataset to classify the test instance using Euclidean distance and weighting function.
  - Assign k = 3. The distance calculation is shown in Table 4.5.

Step 1 : Finding Euclidean distance

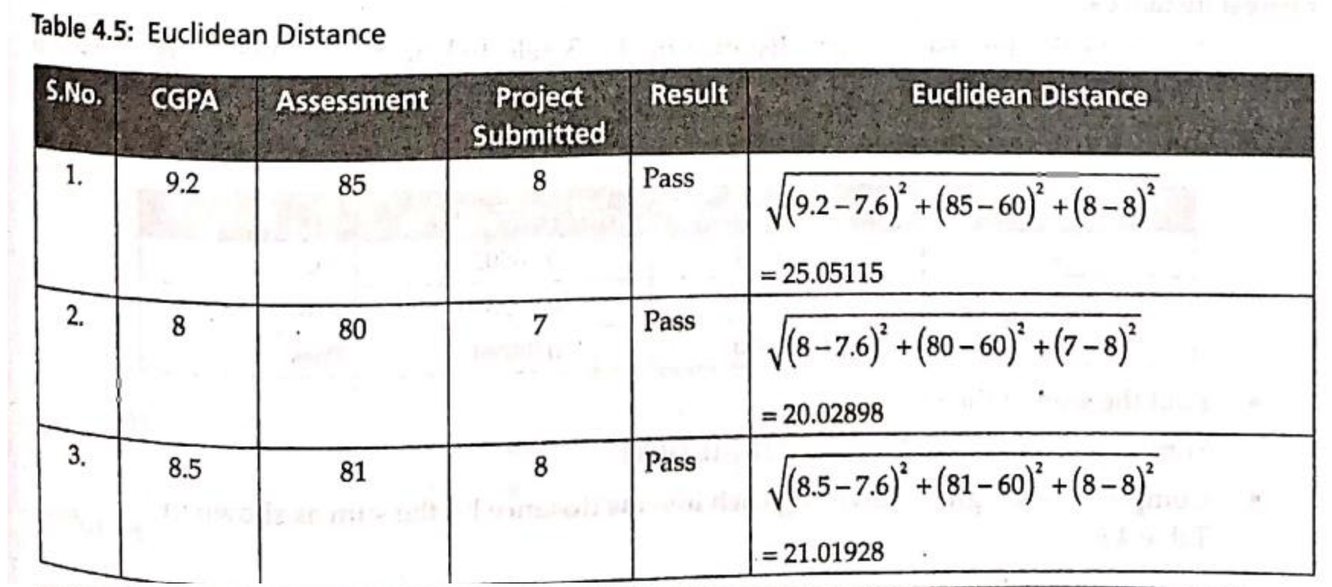
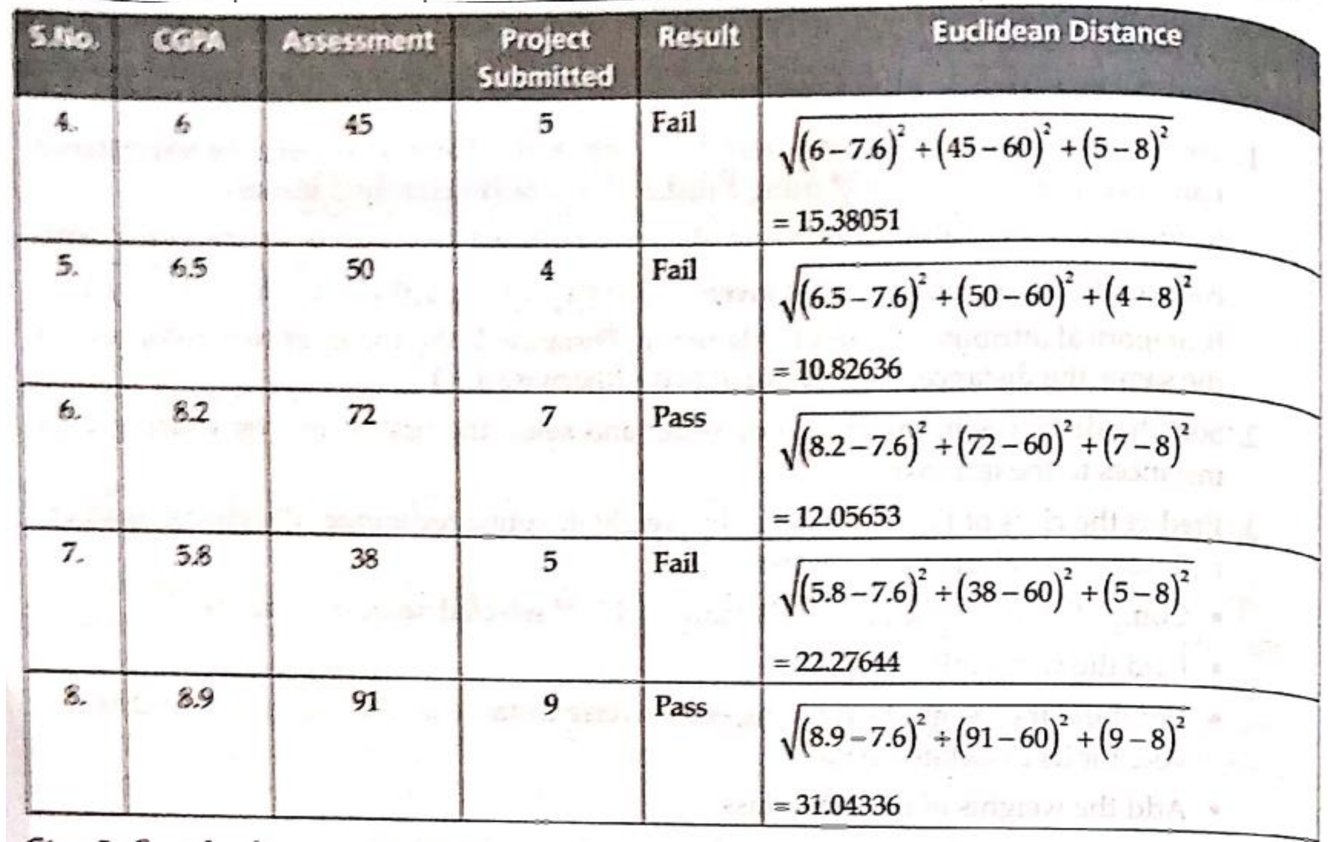
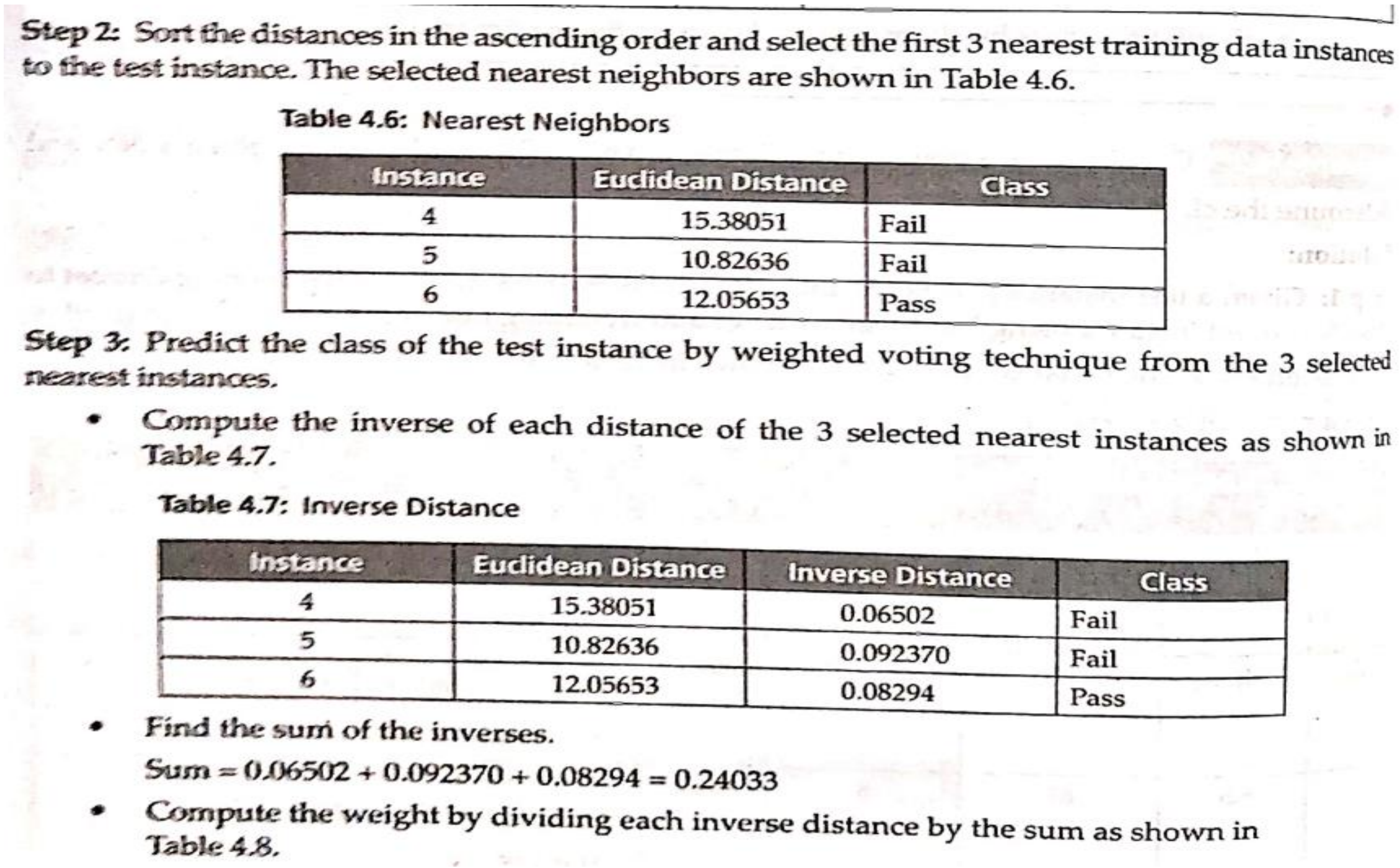



Table 4.5: Euclidean Distance

S.No.	CGPA	Assessment	Project Submitted	Result	Euclidean Distance
1.	9.2	85	8	Pass	$\sqrt{(9.2-7.6)^2+(85-60)^2+(8-8)^2}$ $=25.05115$
2.	8	80	7	Pass	$\sqrt{(8-7.6)^2+(80-60)^2+(7-8)^2}$ $=20.02898$
3.	8.5	81	8	Pass	$\sqrt{(8.5-7.6)^2+(81-60)^2+(8-8)^2}$ $=21.01928$
4.	6	45	5	Fail	$\sqrt{(6-7.6)^2+(45-60)^2+(5-8)^2}$ $=15.38051$
5.	6.5	50	4	Fail	$\sqrt{(6.5-7.6)^2+(50-60)^2+(4-8)^2}$ $=10.82636$
6.	8.2	72	7	Pass	$\sqrt{(8.2-7.6)^2+(72-60)^2+(7-8)^2}$ $=12.05653$
7.	5.8	38	5	Fail	$\sqrt{(5.8-7.6)^2+(38-60)^2+(5-8)^2}$ $=22.27644$
8.	8.9	91	9	Pass	$\sqrt{(8.9-7.6)^2+(91-60)^2+(9-8)^2}$ $=31.04336$

Step 2 and Step 3



Step 2: Sort the distances in the ascending order and select the first 3 nearest training data instances to the test instance. The selected nearest neighbors are shown in Table 4.6.

Table 4.6: Nearest Neighbors

Instance	Euclidean Distance	Class
4	15.38051	Fail
5	10.82636	Fail
6	12.05653	Pass

Step 3: Predict the class of the test instance by weighted voting technique from the 3 selected nearest instances.

- Compute the inverse of each distance of the 3 selected nearest instances as shown in Table 4.7.

Table 4.7: Inverse Distance

Instance	Euclidean Distance	Inverse Distance	Class
4	15.38051	0.06502	Fail
5	10.82636	0.092370	Fail
6	12.05653	0.08294	Pass

- Find the sum of the inverses.

  Sum = 0.06502 + 0.092370 + 0.08294 = 0.24033

- Compute the weight by dividing each inverse distance by the sum as shown in Table 4.8.

Table 4.8: Weight Calculation

Instance	Euclidean Distance	Inverse Distance	Weight = Inverse distance/Sum	Class
4	15.38051	0.06502	0.270545	Fail
5	10.82636	0.092370	0.384347	Fail
6	12.05653	0.08294	0.345109	Pass

- Add the weights of the same class.

  Fail = 0.270545 + 0.384347 = **0.654892**

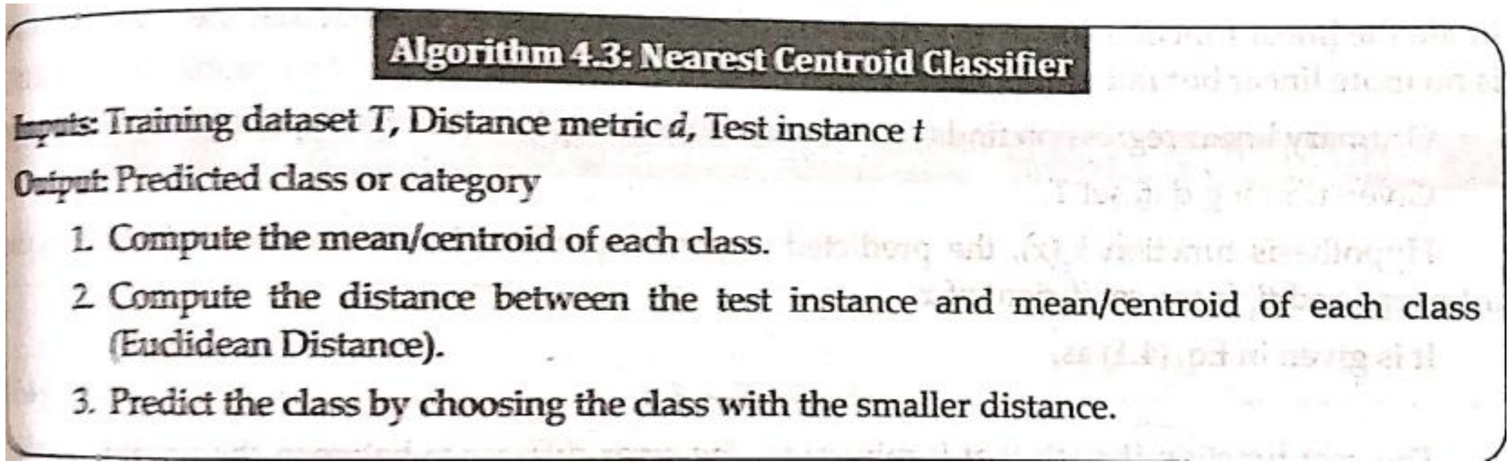
  Pass = 0.345109

- Predict the class by choosing the class with the maximum vote.

The class is predicted as **'Fail'**.

NEAREST CENTROID CLASSIFIER

- A simple alternative to k-NN classifiers for similarity-based classification is the Nearest Centroid Classifier.
- It is a simple classifier and also called as Mean Difference classifier. The idea of this classifier is to classify a test instance to the class whose centroid/mean is closest to that instance.



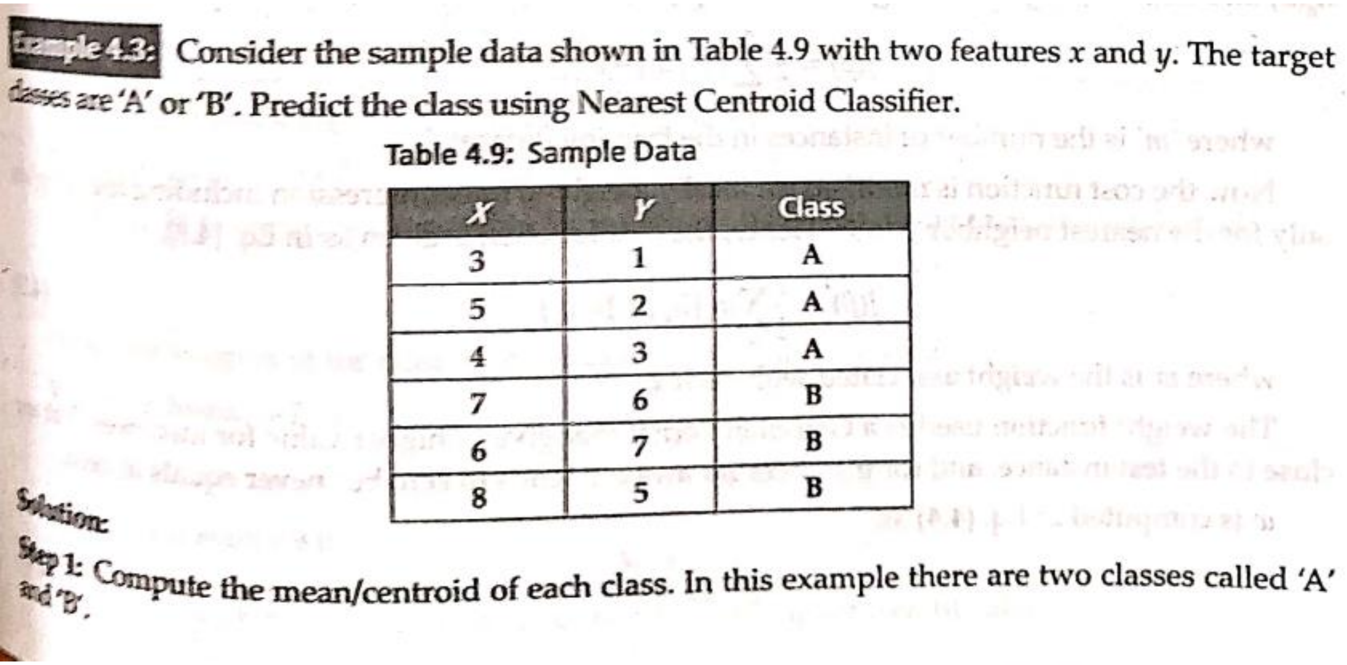
Algorithm 4.3: Nearest Centroid Classifier

Inputs: Training dataset T, Distance metric d, Test instance t

Output: Predicted class or category

```
1. Compute the mean/centroid of each class.
2. Compute the distance between the test instance and mean/centroid of each class
   (Euclidean Distance).
3. Predict the class by choosing the class with the smaller distance.
```

Example



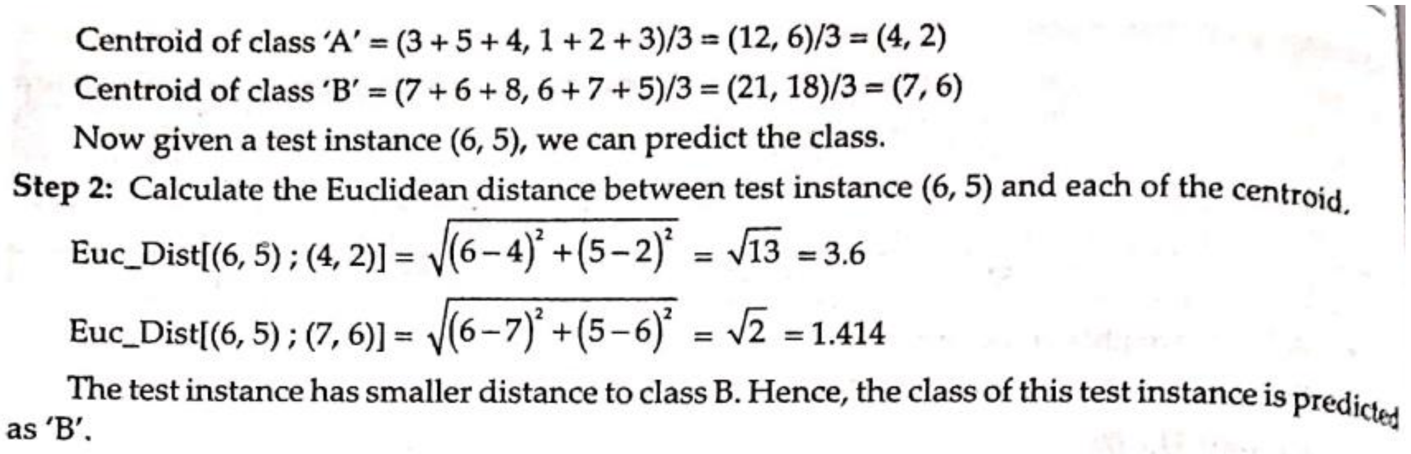
Example 4.3: Consider the sample data shown in Table 4.9 with two features *x* and *y*. The target classes are 'A' or 'B'. Predict the class using Nearest Centroid Classifier.

Table 4.9: Sample Data

x	*y*	Class
3	1	A
5	2	A
4	3	A
7	6	B
6	7	B
8	5	B

Solution:

Step 1: Compute the mean/centroid of each class. In this example there are two classes called 'A' and 'B'.



Centroid of class 'A' = (3 + 5 + 4, 1 + 2 + 3)/3 = (12, 6)/3 = (4, 2)

Centroid of class 'B' = (7 + 6 + 8, 6 + 7 + 5)/3 = (21, 18)/3 = (7, 6)

Now given a test instance (6, 5), we can predict the class.

Step 2: Calculate the Euclidean distance between test instance (6, 5) and each of the centroid.

$$\text{Euc\_Dist}[(6, 5)\,;\,(4, 2)] = \sqrt{(6-4)^2 + (5-2)^2} = \sqrt{13} = 3.6$$

$$\text{Euc\_Dist}[(6, 5)\,;\,(7, 6)] = \sqrt{(6-7)^2 + (5-6)^2} = \sqrt{2} = 1.414$$

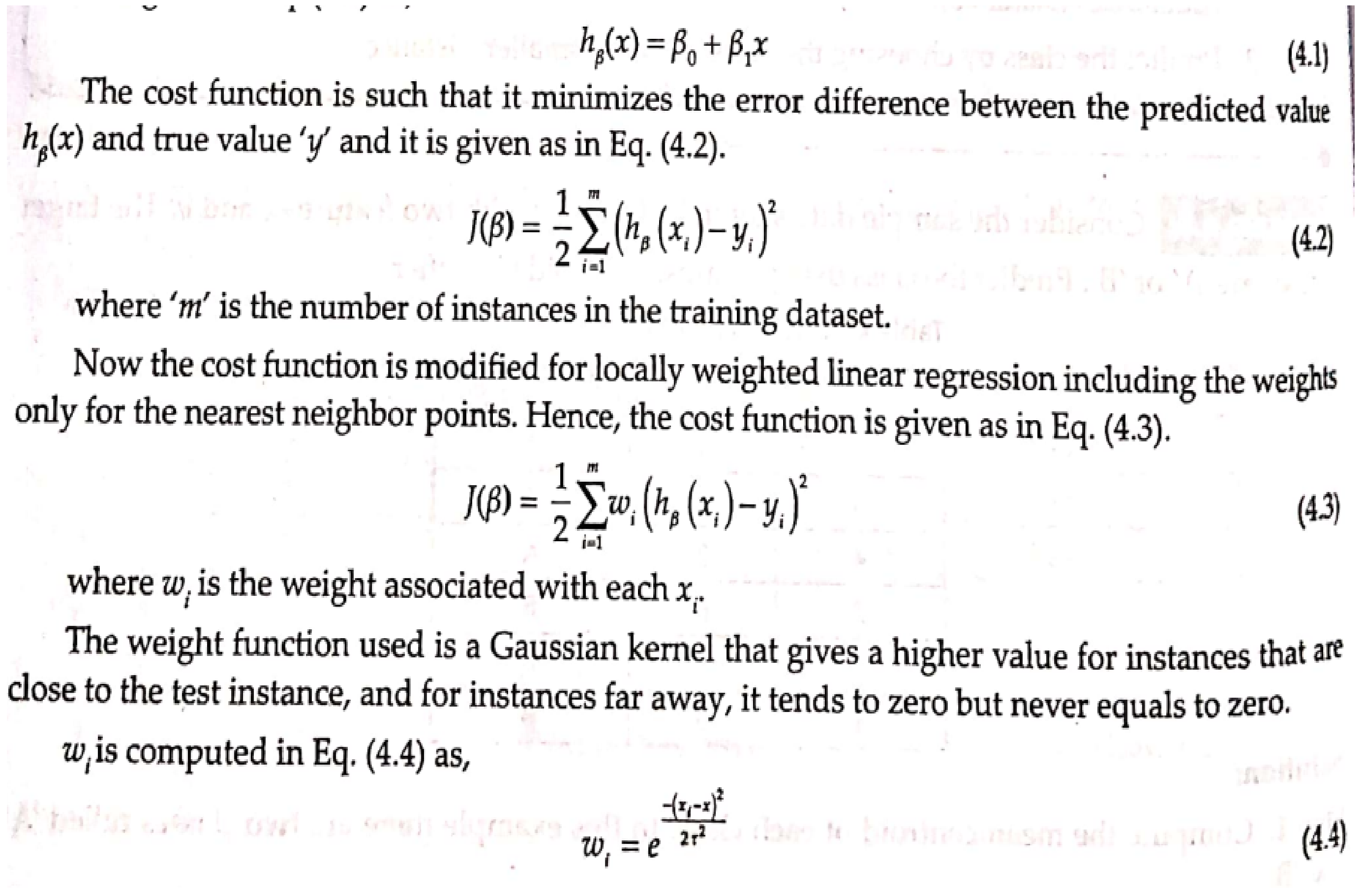
The test instance has smaller distance to class B. Hence, the class of this test instance is predicted as 'B'.

LOCALLY WEIGHTED REGRESSION (LWR)

- Locally Weighted Regression (LWR) is a non-parametric supervised learning algorithm that performs local regression by combining regression model with nearest neighbor's model.
- LWR is also referred to as a memory-based method as it requires training data while prediction but uses only the training data instances locally around the point of interest.
- Using nearest neighbors algorithm, we find the instances that are closest to a test instance and fit linear function to each of those 'k' nearest instances in the local regression model.
- The key idea is that we need to approximate the linear functions of all 'k' neighbors that minimize the error such that the prediction line is no more linear but rather it is a curve.

LOCALLY WEIGHTED REGRESSION (LWR)

- Ordinary linear regression finds out a linear relationship between the input x and the output y. Given training dataset T,
- Hypothesis function h (x), the predicted target output is a linear function where Bo is the intercept and ß1 is the coefficient of x.
- It is given in Eq. (4.1) as,



$$h_\beta(x) = \beta_0 + \beta_1 x \tag{4.1}$$

The cost function is such that it minimizes the error difference between the predicted value $h_\beta(x)$ and true value 'y' and it is given as in Eq. (4.2).

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\beta(x_i) - y_i\right)^2 \tag{4.2}$$

where 'm' is the number of instances in the training dataset.

Now the cost function is modified for locally weighted linear regression including the weights only for the nearest neighbor points. Hence, the cost function is given as in Eq. (4.3).

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m} w_i\left(h_\beta(x_i) - y_i\right)^2 \tag{4.3}$$

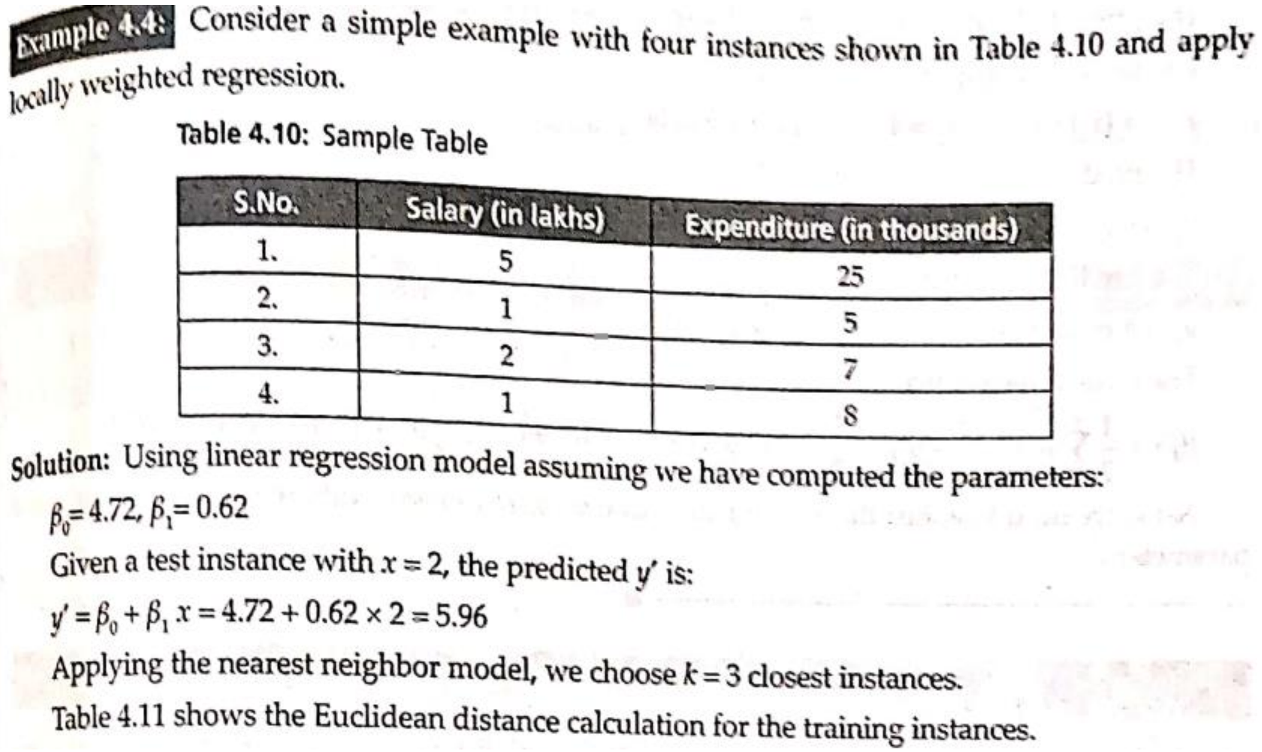
where w_i is the weight associated with each x_i.

The weight function used is a Gaussian kernel that gives a higher value for instances that are close to the test instance, and for instances far away, it tends to zero but never equals to zero.

w_i is computed in Eq. (4.4) as,

$$w_i = e^{\frac{-(x_i - x)^2}{2\tau^2}} \tag{4.4}$$

Example



Example 4.4: Consider a simple example with four instances shown in Table 4.10 and apply locally weighted regression.

Table 4.10: Sample Table

S.No.	Salary (in lakhs)	Expenditure (in thousands)
1.	5	25
2.	1	5
3.	2	7
4.	1	8

Solution: Using linear regression model assuming we have computed the parameters:

$\beta_0 = 4.72, \beta_1 = 0.62$

Given a test instance with $x = 2$, the predicted y' is:

$y' = \beta_0 + \beta_1 x = 4.72 + 0.62 \times 2 = 5.96$

Applying the nearest neighbor model, we choose $k = 3$ closest instances.

Table 4.11 shows the Euclidean distance calculation for the training instances.

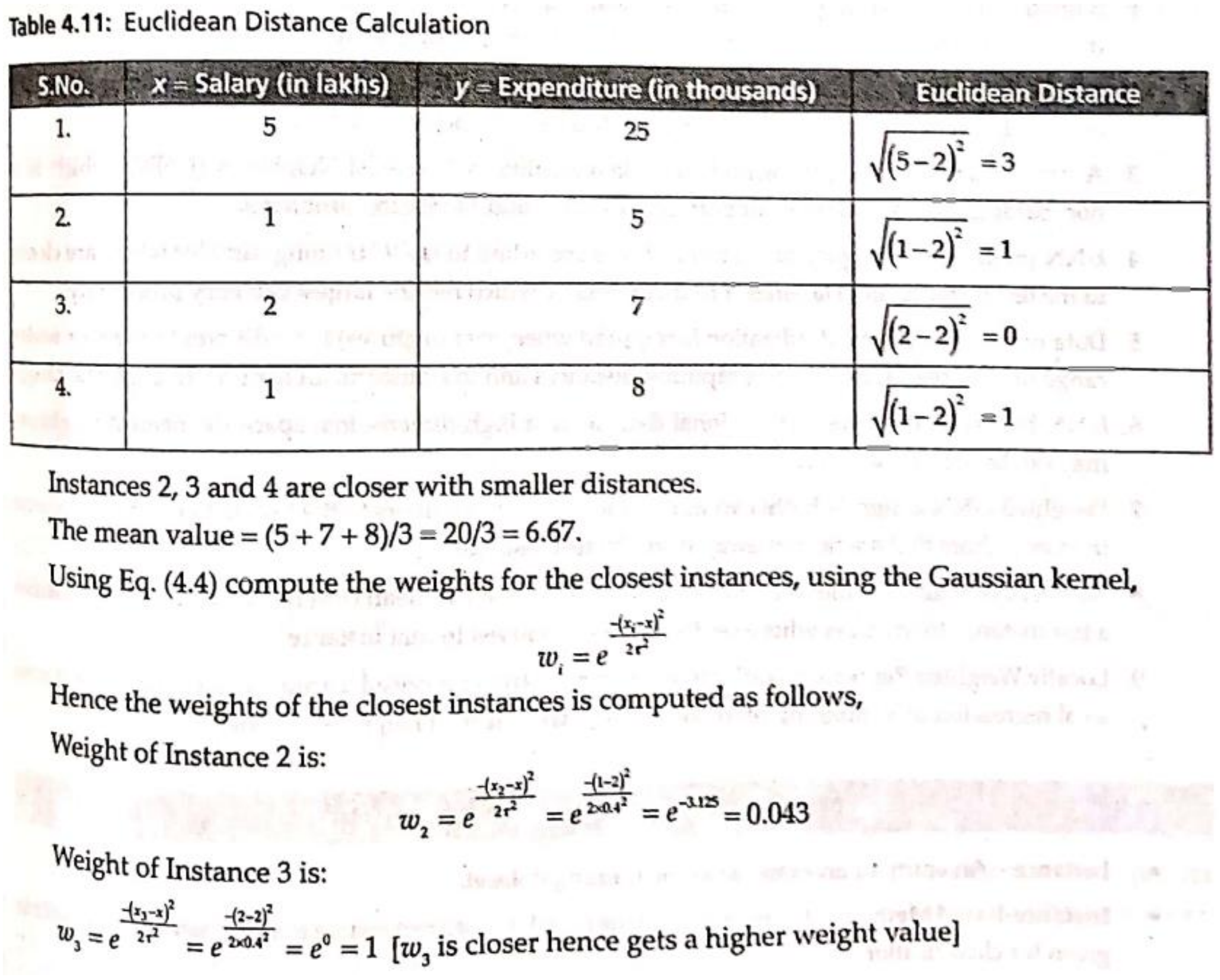


Table 4.11: Euclidean Distance Calculation

S.No.	x = Salary (in lakhs)	y = Expenditure (in thousands)	Euclidean Distance
1.	5	25	$\sqrt{(5-2)^2} = 3$
2.	1	5	$\sqrt{(1-2)^2} = 1$
3.	2	7	$\sqrt{(2-2)^2} = 0$
4.	1	8	$\sqrt{(1-2)^2} = 1$

Instances 2, 3 and 4 are closer with smaller distances.

The mean value = (5 + 7 + 8)/3 = 20/3 = 6.67.

Using Eq. (4.4) compute the weights for the closest instances, using the Gaussian kernel,

$$w_i = e^{\frac{-(x_i - x)^2}{2\tau^2}}$$

Hence the weights of the closest instances is computed as follows,

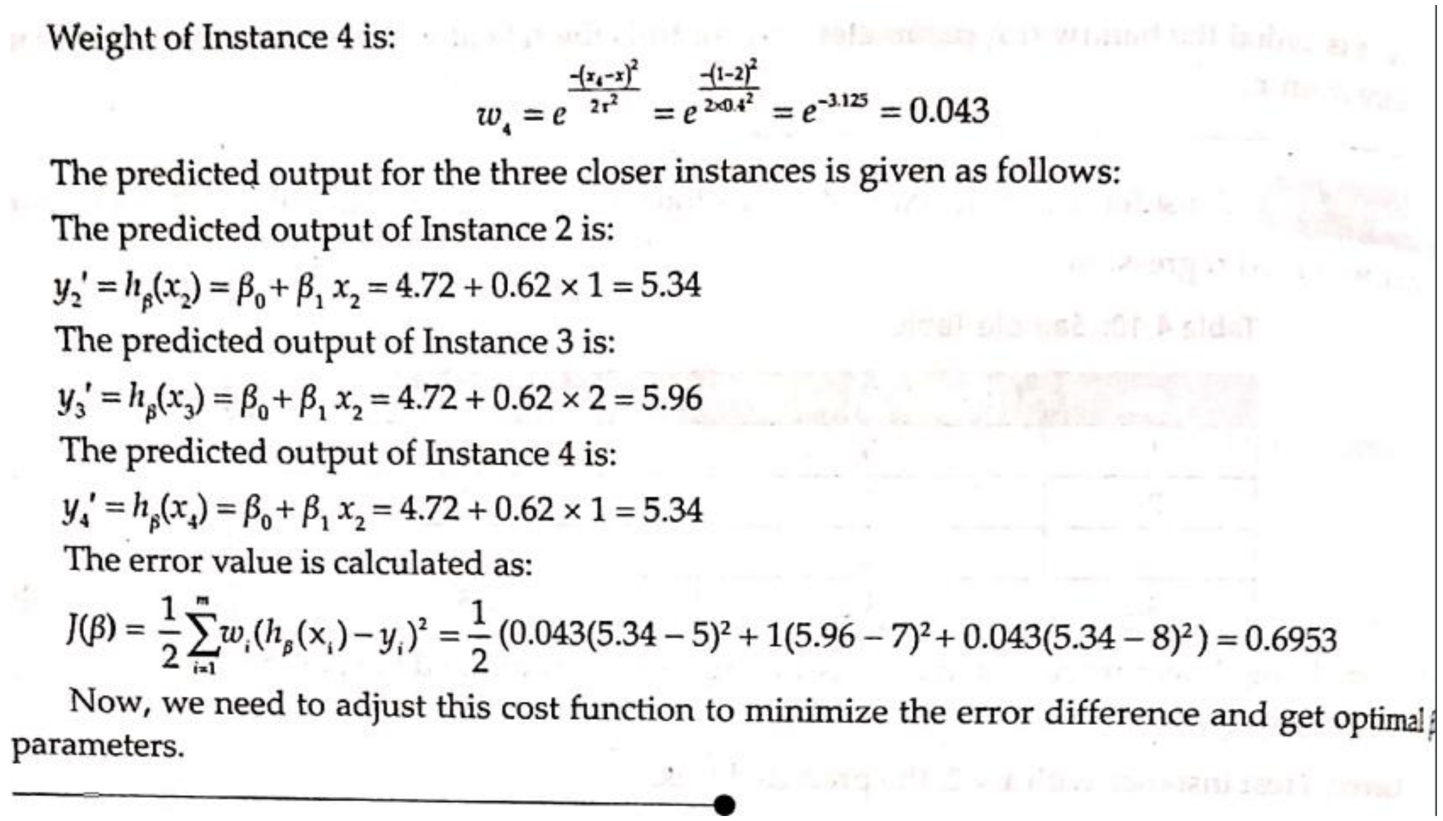
Weight of Instance 2 is:

$$w_2 = e^{\frac{-(x_2 - x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2\times 0.4^2}} = e^{-3.125} = 0.043$$

Weight of Instance 3 is:

$w_3 = e^{\frac{-(x_3 - x)^2}{2\tau^2}} = e^{\frac{-(2-2)^2}{2\times 0.4^2}} = e^0 = 1$ [w_3 is closer hence gets a higher weight value]

Example



Weight of Instance 4 is:

$$w_4 = e^{\frac{-(x_4 - x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2\times0.4^2}} = e^{-3.125} = 0.043$$

The predicted output for the three closer instances is given as follows:

The predicted output of Instance 2 is:

$y_2' = h_\beta(x_2) = \beta_0 + \beta_1\, x_2 = 4.72 + 0.62 \times 1 = 5.34$

The predicted output of Instance 3 is:

$y_3' = h_\beta(x_3) = \beta_0 + \beta_1\, x_2 = 4.72 + 0.62 \times 2 = 5.96$

The predicted output of Instance 4 is:

$y_4' = h_\beta(x_4) = \beta_0 + \beta_1\, x_2 = 4.72 + 0.62 \times 1 = 5.34$

The error value is calculated as:

$$J(\beta) = \frac{1}{2}\sum_{i=1}^{m} w_i (h_\beta(x_i) - y_i)^2 = \frac{1}{2}(0.043(5.34-5)^2 + 1(5.96-7)^2 + 0.043(5.34-8)^2) = 0.6953$$

Now, we need to adjust this cost function to minimize the error difference and get optimal parameters.

Regression Analysis

Module 3

Chapter 3

INTRODUCTION TO REGRESSION

- Regression analysis is the premier method of supervised learning.
- This is one of the most popular and oldest supervised learning technique.
- Given a training dataset D containing N training points(x, y,), where i = 1 ... N, regression analysis is used to model the relationship between one or more independent variables x, and a dependent variable y,
- The relationship between the dependent and independent variables can be represented as a function as follows: $y=f(x)$
- Here, the feature variable x is also known as an explanatory variable, exploratory variable, a predictor variable, an independent variable, a covariate, or a domain point. y is a dependent variable.
- Dependent variables are also called as labels, target variables, or response variables.

INTRODUCTION TO REGRESSION

- Regression analysis determines the change in response variables when one exploration variable is varied while keeping all other parameters constant.
- This is used to determine the relationship each of the exploratory variables exhibits. Thus, regression analysis is used for prediction and forecasting.
- Regression is used to predict continuous variables or quantitative variables such as price and revenue.
- Thus, the primary concern of regression analysis is to find answer to questions such as:
  - What is the relationship between the variables?
  - What is the strength of the relationships?
  - What is the nature of the relationship such as linear or non-linear?
  - What is the relevance of the attributes?
  - What is the contribution of each attribute?

INTRODUCTION TO REGRESSION

- There are many applications of regression analysis. Some of the applications of regressions
  - Sales of a goods or services
  - Value of bonds in portfolio management
  - Premium on insurance companies
  - Yield of crops in agriculture
  - Prices of real estate

Introduction To Linearity, Correlation, And Causation

- The quality of the regression analysis is determined by the factors such as correlation and causation.
- Regression and Correlation
  - Correlation among two variables can be done effectively using a Scatter plot, which is a plot between explanatory variables and response variables.
  - It is a 2D graph showing the relationship between two variables.
  - The x-axis of the scatter plot is independent, or input or predictor variables and y-axis of the scatter plot is output or dependent or predicted variables.
  - The scatter plot is useful in exploring data. Some of the scatter plots are shown in Figure 5.1.
  - The Pearson correlation coefficient is the most common test for determining correlation if there is an association between two variables.
  - The correlation coefficient is denoted by r.
  - The positive, negative, and random correlations are given in Figure 5.1.
    - In positive correlation, one variable change is associated with the change in another variable.
    - In negative correlation, the relationship between the variables is reciprocal while in random correlation, no relationship exists between variables.

Introduction To Linearity, Correlation, And Causation

- While correlation is about relationships among variables, say x and y, regression is about predicting one variable given another variable.

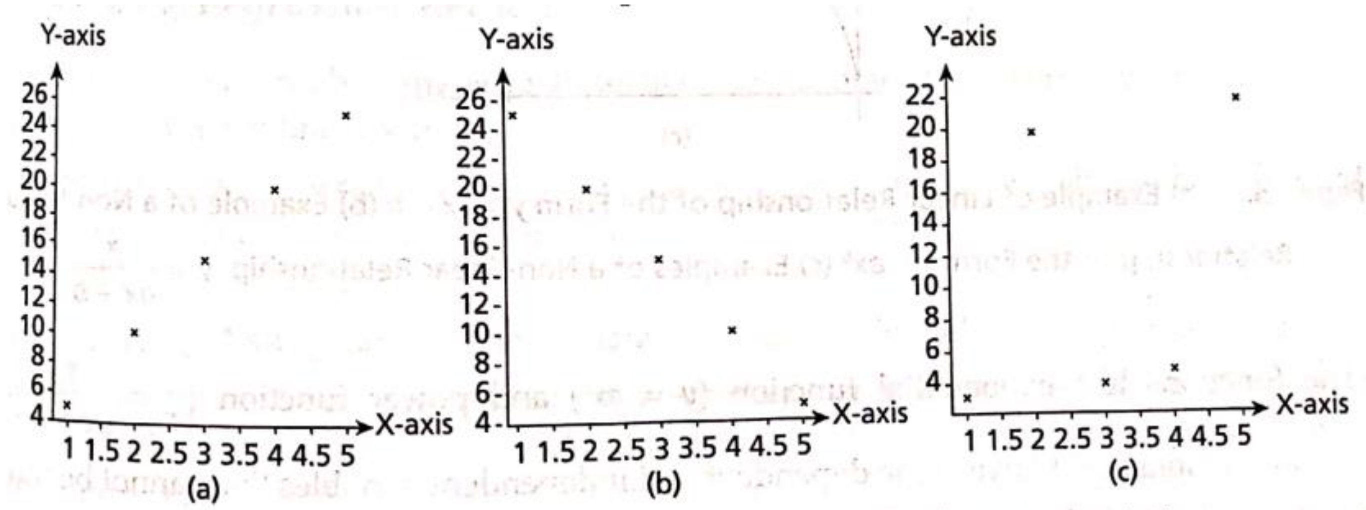


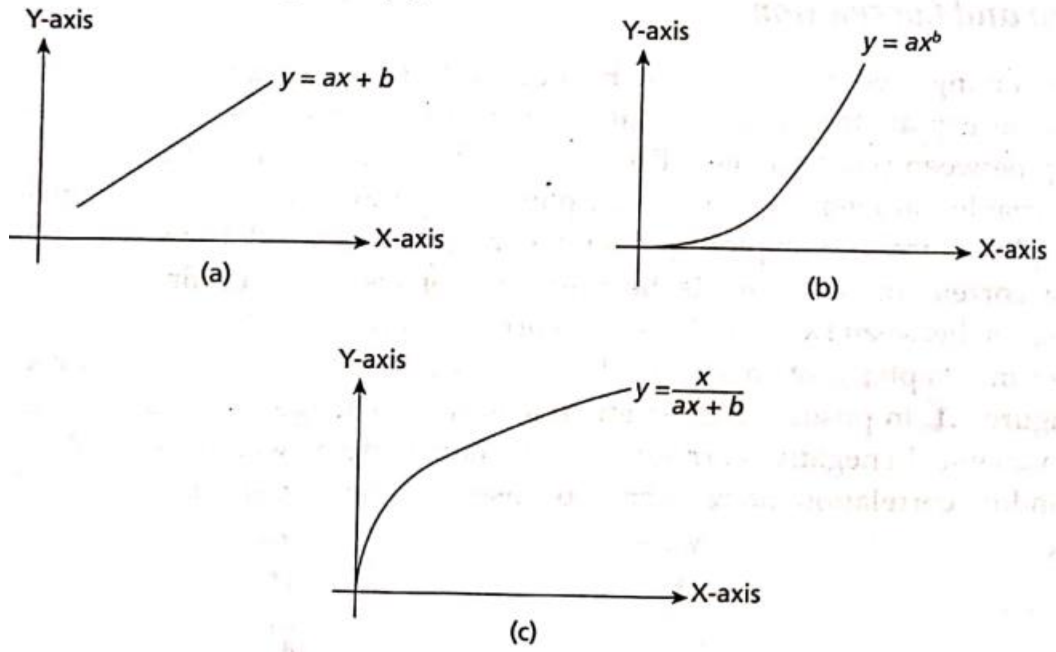
Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation (c) Random Points with No Correlation

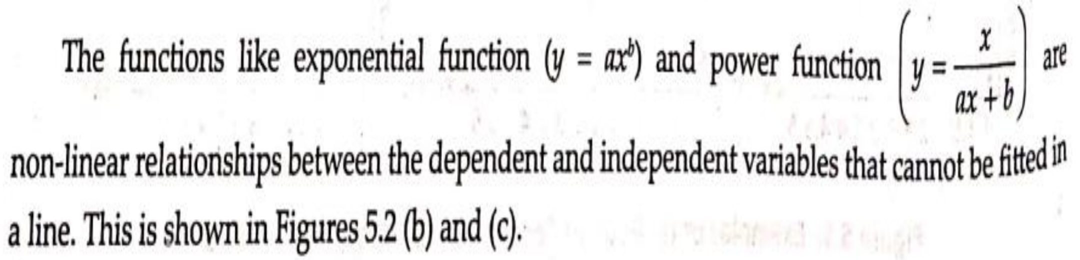
Introduction To Linearity, Correlation, And Causation

- Regression and Causation
  - Causation is about causal relationship among variables, say x and y. Causation means know in whether x causes y to happen or vice versa. x causes y is often denoted as x implies y.
  - Correlation and Regression relationships are not same as causation relationship. For example, the correlation between economical background and marks scored does not imply that economic background causes high marks. Similarly, the relationship between higher sales of cool drinks due to a raise in temperature is not a causal relation. Even though high temperature is the cause of cool drinks sales, it depends on other factors too.

Introduction To Linearity, Correlation, And Causation

- Linearity and Non-linearity Relationships
  - The linearity relationship between the variables means the relationship between the dependent and independent variables can be visualized as a straight line.
  - The line of the form, y = ax + b can be fitted to the data points that indicate the relationship between x and y. By linearity, it is meant that as one variable increases, the corresponding variable also increases in a linear manner.
  - A linear relationship is shown in Figure 5.2 (a). A non-linear relationship exists in functions such as exponential function and power function and it is shown in Figures 5.2 (b) and 5.2 (c). Here, x-axis is given by x data and y-axis is given by y data.




The functions like exponential function ($y = ax^b$) and power function $\left(y = \frac{x}{ax+b}\right)$ are non-linear relationships between the dependent and independent variables that cannot be fitted in a line. This is shown in Figures 5.2 (b) and (c).

Types of Regression Methods

- Linear Regression
  - It is a type of regression where a line is fitted upon given data for finding the linear relationship between one independent variable and one dependent variable to describe relationships.
- Multiple Regression
  - It is a type of regression where a line is fitted for finding the linear relationship between two or more independent variables and one dependent variable to describe relationships among variables.

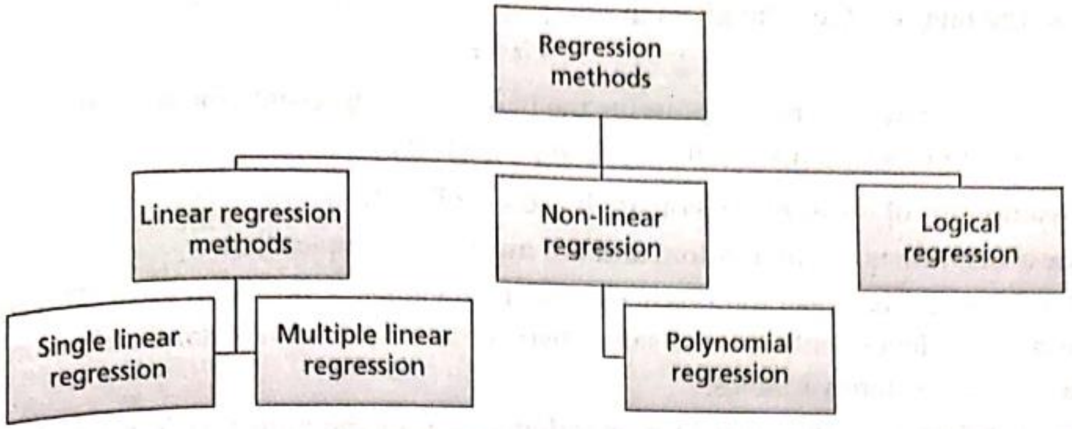


Figure 5.3: Types of Regression Methods

Types of Regression Methods

- Polynomial Regression
  - It is a type of non-linear regression method of describing relation-ships among variables where Nth degree polynomial is used to model the relationship between one independent variable and one dependent variable. Polynomial multiple regression is used to model two or more independent variables and one dependent variable.
- Logistic Regression
  - It is used for predicting categorical variables that involve one or more independent variables and one dependent variable. This is also known as a binary classifier.
- Lasso and Ridge Regression Methods
  - These are special variants of regression method where regularization methods are used to limit the number and size of coefficients of the independent variables.

Limitations of Regression Method

- Outliers-Outliers are abnormal data. It can bias the outcome of the regression model, as outliers push the regression line towards it.
- Number of cases - The ratio of independent and dependent variables should be at least 20 : 1.For every explanatory variable, there should be at least 20 samples. Atleast five samples are required in extreme cases.
- Missing data - Missing data in training data can make the model unfit for the sampled data.
- Multicollinearity - If exploratory variables are highly correlated (0.9 and above), the regression is vulnerable to bias. Singularity leads to perfect correlation of 1. The remedy is to remove exploratory variables that exhibit correlation more than 1. If there is a tie, then the tolerance(1 -R squared) is used to eliminate variables that have the greatest value.

INTRODUCTION TO LINEAR REGRESSION

- In the simplest form, the linear regression model can be created by fitting a line among the scattered data points.
- The line is of the form given in Eq. (5.2).

$$y = a_0 + a_1 \times x + e$$

- Here, a, is the intercept which represents the bias and a, represents the slope of the line. These are called regression coefficients. e is the error in prediction.
- The assumptions of linear regression are listed as follows:
  - The observations (y) are random and are mutually independent.
  - The difference between the predicted and true values is called an error. The error is alsmutually independent with the same distributions such as normal distribution with zeromean and constant variables.

INTRODUCTION TO LINEAR REGRESSION

- The assumptions of linear regression are listed as follows:
  - The observations (y) are random and are mutually independent.
  - The difference between the predicted and true values is called an error. The error is also mutually independent with the same distributions such as normal distribution with zero mean and constant variables.
  - The distribution of the error term is independent of the joint distribution of explanatory variables.
  - The unknown parameters of the regression models are constants.
- The idea of linear regression is based on Ordinary Least Square (OLS) approach.
- This method is also known as ordinary least squares method. In this method, the data points are modelled using a straight line.

INTRODUCTION TO LINEAR REGRESSION

- Any arbitrarily drawn line is not an optimal line.
- In Figure 5.4, three datapoints and their errors (e1, e2, e3) are shown.
- The vertical distance between each point and the line (predicted by the approximate line equation y = a0 + a1x) is called an error.
- These individual errors are added to compute the total error of the predicted line. This is called sum of residuals.
- The squares of the individual errors can also be computed and added to give a sum of squared error.
- The line with the lowest sum of squared error is called line of best fit.

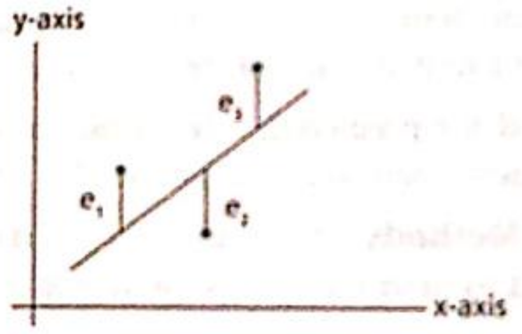
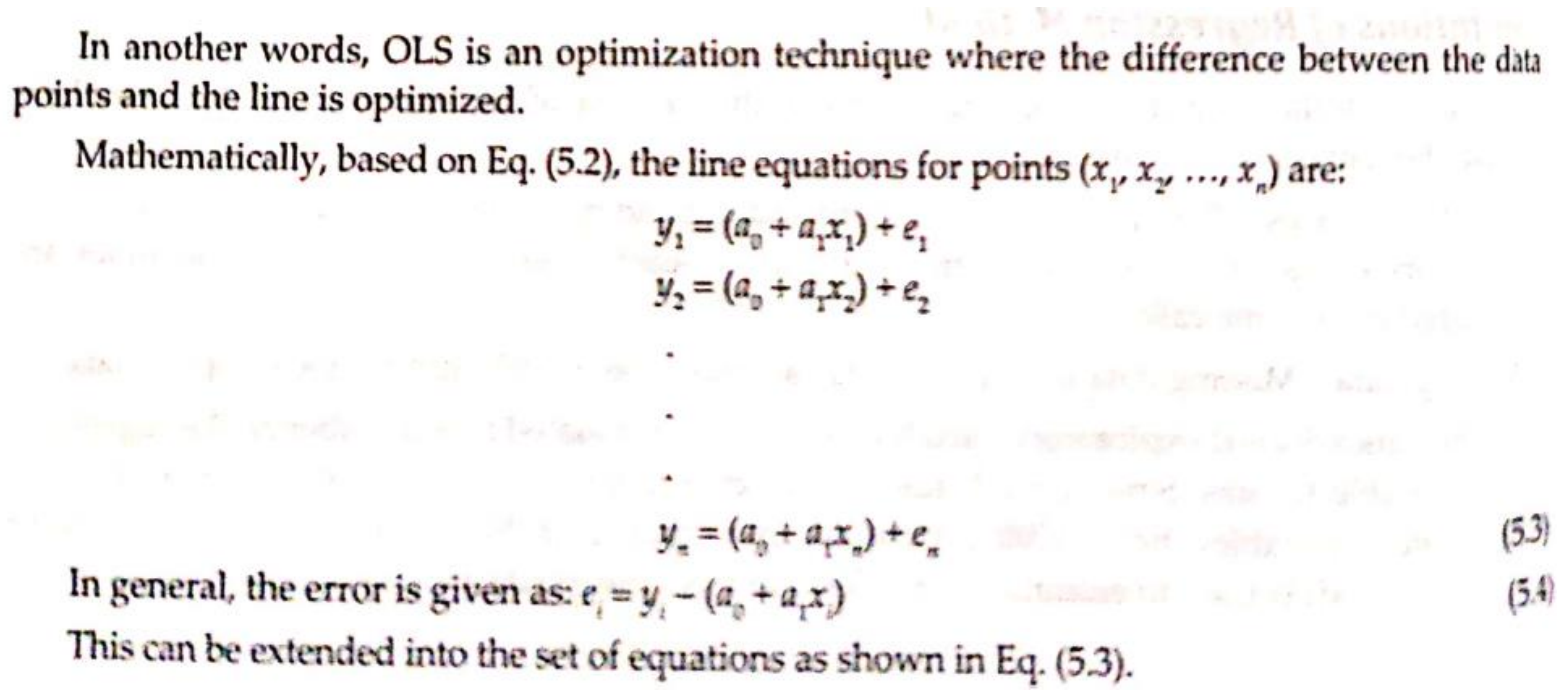


Figure 5.4: Data Points and their Errors

INTRODUCTION TO LINEAR REGRESSION



In another words, OLS is an optimization technique where the difference between the data points and the line is optimized.

Mathematically, based on Eq. (5.2), the line equations for points $(x_1, x_2, \ldots, x_n)$ are:

$$y_1 = (a_0 + a_1x_1) + e_1$$
$$y_2 = (a_0 + a_1x_2) + e_2$$
$$\vdots$$
$$y_n = (a_0 + a_1x_n) + e_n \tag{5.3}$$

In general, the error is given as: $e_i = y_i - (a_0 + a_1x_i)$ (5.4)

This can be extended into the set of equations as shown in Eq. (5.3).

INTRODUCTION TO LINEAR REGRESSION

- Here, the terms (e1, e2, ... , e,) are error associated with the data points and denote the difference between the true value of the observation and the point on the line. This is also called as residuals. The residuals can be positive, negative or zero.
- A regression line is the line of best fit for which the sum of the squares of residuals is minimum. The minimization can be done as minimization of individual errors by finding the parameters a0 and a1 such that:

$$E = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))$$

- Or as the minimization of sum of absolute values of the individual errors:

values of the indiv

$$E = \sum_{i=1}^{n} \left| e_i \right| = \sum_{i=1}^{n} \left| (y_i - (a_0 + a_1 x_i)) \right|$$

- Or as the minimization of the sum of the squares of the individual errors:

$$E = \sum_{i=1}^{n} (e_i)^2 = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))^2$$

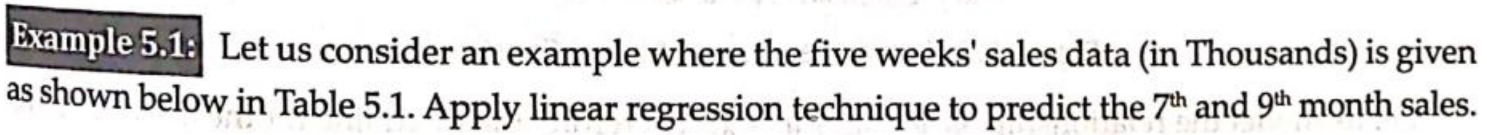
INTRODUCTION TO LINEAR REGRESSION

- Sum of the squares of the individual errors, often preferred as individual errors (positive and negative errors), do not get cancelled out and are always positive, and sum of squares results in a large increase even for a small change in the error. Therefore, this is preferred for linear regression.
- Therefore, linear regression is modelled as a minimization function as follows:

$$J(a_1, a_0) = \sum_{i=1}^{n}[y_i - f(x_i)]^2 = \sum_{i=1}^{n}[y_i - (a_0 + a_1x_i)]^2$$

- Here, J(ao a1) is the criterion function of parameters a, and a. This needs to be minimized.This is done by differentiating and substituting to zero. This yields the coefficient values ofa, and a1. The values of estimates of a, and a, are given as follows: $a_1 = \frac{(\overline{xy}) - (\overline{x})(\overline{y})}{(\overline{x_i^2}) - (\overline{x})^2}$
- And the value of a, is given as follows: $a_0 = (\overline{y}) - a_1 \times \overline{x}$

Example



Example 5.1: Let us consider an example where the five weeks' sales data (in Thousands) is given as shown below in Table 5.1. Apply linear regression technique to predict the 7th and 9th month sales.

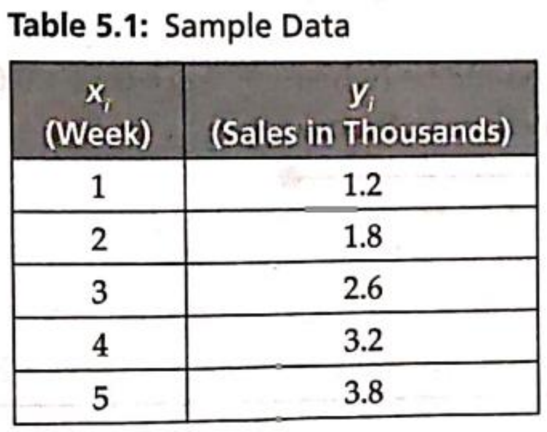


Table 5.1: Sample Data

x_i (Week)	y_i (Sales in Thousands)
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

Solution: Here, there are 5 items, i.e., i = 1, 2, 3, 4, 5. The computation table is shown below (Table 5.2). Here, there are five samples, so i ranges from 1 to 5.

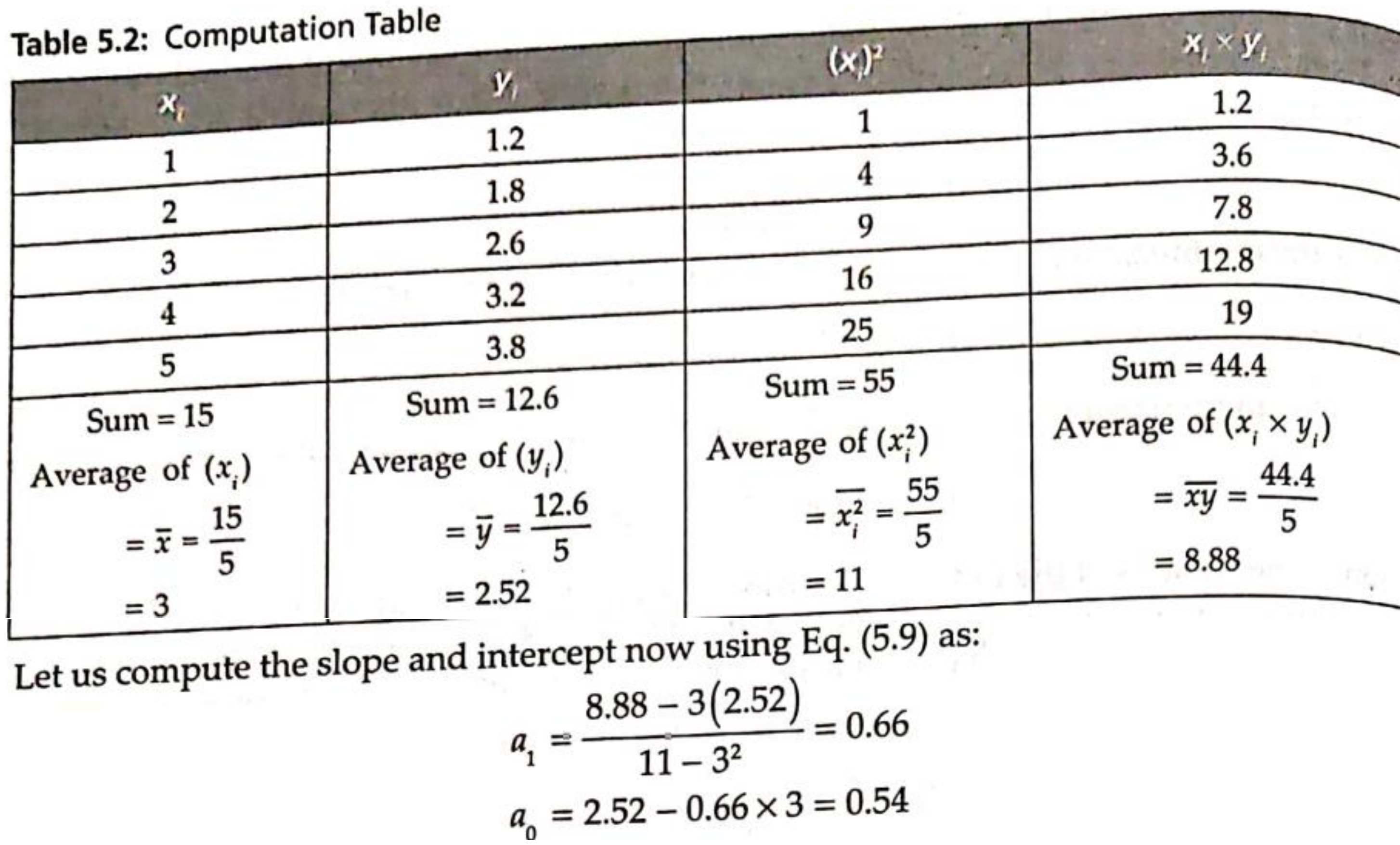


Table 5.2: Computation Table

x_i	y_i	$(x_i)^2$	$x_i \times y_i$
1	1.2	1	1.2
2	1.8	4	3.6
3	2.6	9	7.8
4	3.2	16	12.8
5	3.8	25	19
Sum = 15 Average of (x_i) $= \bar{x} = \frac{15}{5}$ $= 3$	Sum = 12.6 Average of (y_i) $= \bar{y} = \frac{12.6}{5}$ $= 2.52$	Sum = 55 Average of (x_i^2) $= \overline{x_i^2} = \frac{55}{5}$ $= 11$	Sum = 44.4 Average of $(x_i \times y_i)$ $= \overline{xy} = \frac{44.4}{5}$ $= 8.88$

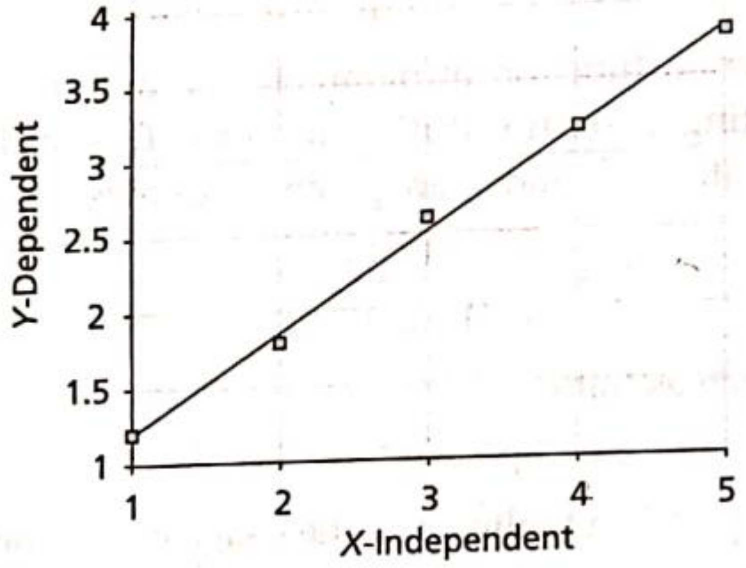
Let us compute the slope and intercept now using Eq. (5.9) as:

$$a_1 = \frac{8.88 - 3(2.52)}{11 - 3^2} = 0.66$$

$$a_0 = 2.52 - 0.66 \times 3 = 0.54$$

Example Cont..

The fitted line is shown in Figure 5.5.



—Regression line ($\hat{y} = 0.66x + 0.54$)

Figure 5.5: Linear Regression Model Constructed

Let us model the relationship as $y = a_0 + a_1 \times x$. Therefore, the fitted line for the above data is: $y = 0.54 + 0.66 \times x$.

The predicted 7th week sale would be (when $x = 7$), $y = 0.54 + 0.66 \times 7 = 5.16$ and the 12th month, $y = 0.54 + 0.66 \times 12 = 8.46$. All sales are in thousands.

Linear Regression in Matrix Form

- Matrix notations can be used for representing the values of independent and dependent variables. This is illustrated through Example 5.2.
- The Eq. (5.3) can be written in the form of matrix as follows:

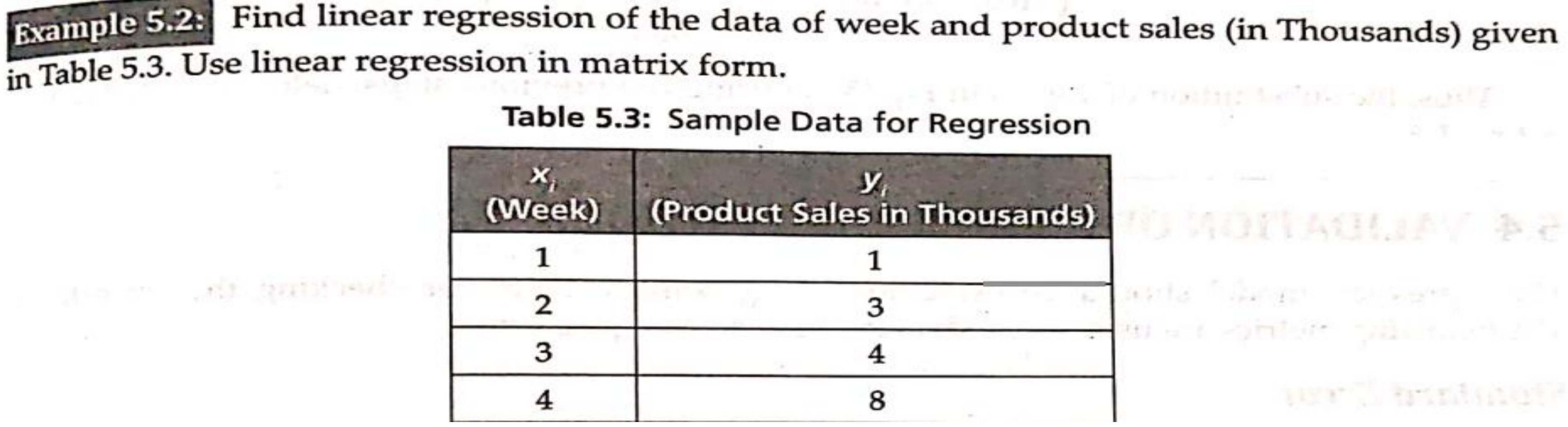
$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

- This can be written as:

$$Y=Xa+e$$

  - where X is an n x 2 matrix, Y is an n x 1 vector, a is a 2 x 1 column vector and e is an n x 1 column vector.

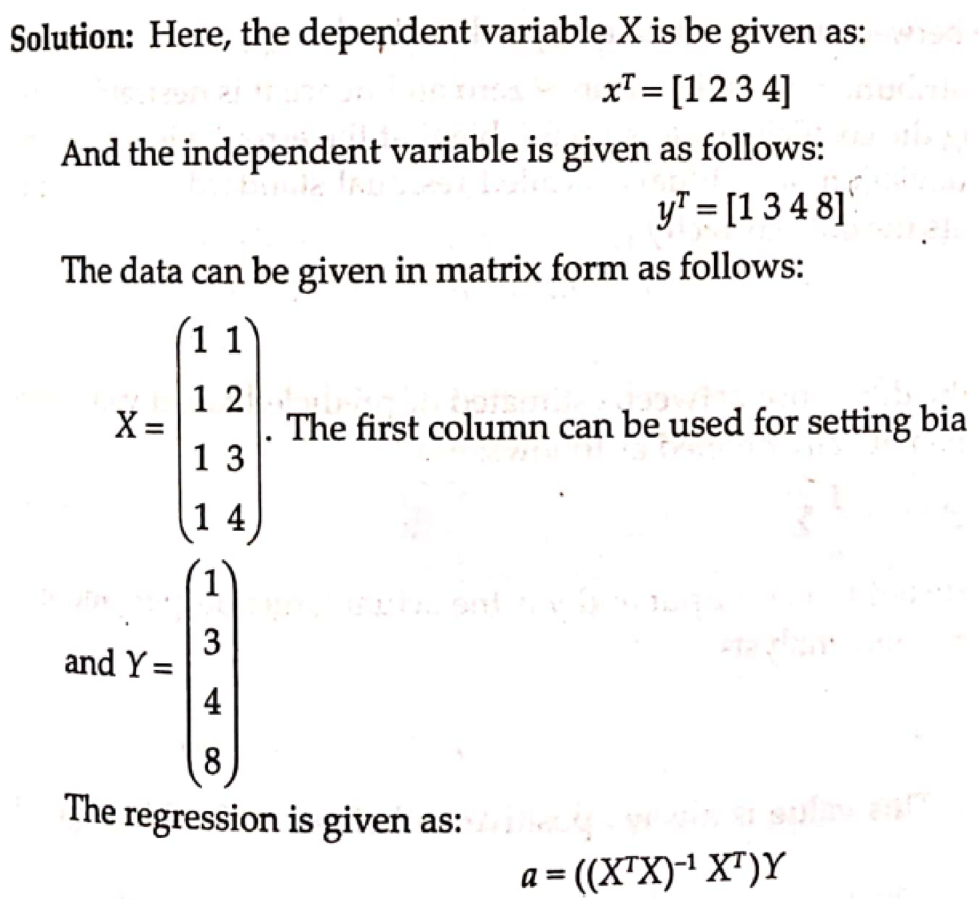
Example



Example 5.2: Find linear regression of the data of week and product sales (in Thousands) given in Table 5.3. Use linear regression in matrix form.

Table 5.3: Sample Data for Regression

x_i (Week)	y_i (Product Sales in Thousands)
1	1
2	3
3	4
4	8



Solution: Here, the dependent variable X is be given as:

$$x^T = [1\ 2\ 3\ 4]$$

And the independent variable is given as follows:

$$y^T = [1\ 3\ 4\ 8]$$

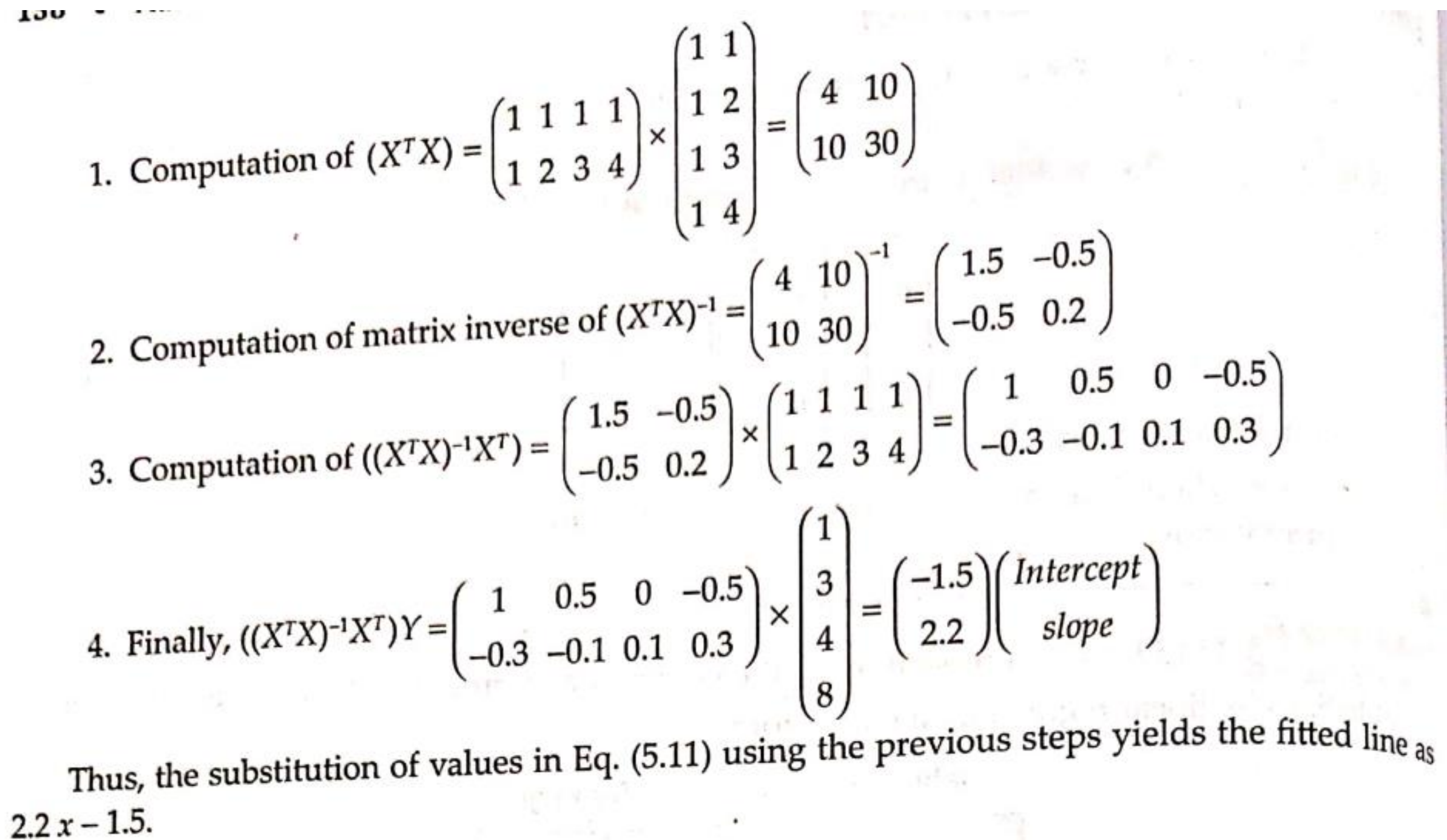
The data can be given in matrix form as follows:

$X = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix}$. The first column can be used for setting bia

and $Y = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix}$

The regression is given as:

$$a = ((X^T X)^{-1} X^T)Y$$



1. Computation of $(X^T X) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$

2. Computation of matrix inverse of $(X^T X)^{-1} = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}^{-1} = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix}$

3. Computation of $((X^T X)^{-1} X^T) = \begin{pmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix}$

4. Finally, $((X^T X)^{-1} X^T)Y = \begin{pmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 3 \\ 4 \\ 8 \end{pmatrix} = \begin{pmatrix} -1.5 \\ 2.2 \end{pmatrix} \begin{pmatrix} \textit{Intercept} \\ \textit{slope} \end{pmatrix}$

Thus, the substitution of values in Eq. (5.11) using the previous steps yields the fitted line as $2.2\,x - 1.5$.

5.4 VALIDATION OF REGRESSION METHODS

- The regression model should be evaluated using some metrics for checking the correctness. The following metrics are used to validate the results of regression.
- Standard Error
  - Residuals or error is the difference between the actual (y) and predicted value (ŷ).If the residuals have normal distribution, then the mean is zero and hence it is desirable. This is a measure of variability in finding the coefficients. It is preferable that the error be less than the coefficient estimate. The standard deviation of residuals is called residual standard error. If it is zero, then it means that the model fits the data correctly.

5.4 VALIDATION OF REGRESSION METHODS

- Mean Absolute Error (MAE)
  - MAE is the mean of residuals. It is the difference between estimated or predicted target value and actual target incomes. It can be mathematically defined as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=0}^{n-1}\left|y_i - \hat{y}_i\right|$$

  - Here, y is the estimated or predicted target output and y is the actual target output, and n is the number of samples used for regression analysis.

VALIDATION OF REGRESSION METHODS

- Mean Squared Error (MSE)
  - It is the sum of square of residuals. This value is always positive and closer to 0. This is given mathematically as:

$$\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2$$

- Root Mean Square Error (RMSE)
- The square root of the MSE is called RMSE. This is given as:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}$$

VALIDATION OF REGRESSION METHODS

- Relative MSE
  - Relative MSE is the ratio of the prediction ability of the y to the average of the trivial population. The value of zero indicates that the model is perfect and its value ranges between 0 and 1. If the value is more than 1, then the created model is not a good one. This is given as follows:

$$\text{RelMSE} = \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \overline{y}_i)^2}$$

- Coefficient of Variation
  - Coefficient of variation is unit less and is given as:

$$\text{CV} = \frac{\text{RMSE}}{\overline{y}}$$

Assignment Questions

- Module - 3
- Explain different types of learning
- What is learning system? Explain the steps involved in designing a learning system.
- Linear regression matrix form
- 4.1 example
- Generalization and specialization
- Module -2
- Elements of data
- Data preprocessing
- Data visualization
- problems