Module – 03

Chapter: - 01 - Basics of Learning Theory

- Learning is a process by which one can acquire knowledge and construct new ideas or concepts based on the experiences.
- Machine learning is an intelligent way of learning general concept from training examples without writing a program.
- There are many machine learning algorithms through which computers can intelligently learn from past data or experiences, identify patterns, and make predictions when new data is fed.

Learning Objectives

- Understand the basics of learning theory and the key elements of machine learning.
- Introduce Concept learning to obtain an abstraction and generalization from the data.

  Learn about hypothesis representation language and to explore searching the hypothesis space using Find-S algorithm and its limitations.
- Study about version spaces, List-Then-Eliminate algorithm and Candldate Elimination algorithm. Introduce Inductive bias, a set of prior assumptions considered by a learning algorithm beyond the training data in order to perform induction.
- Discuss the tradeoff between the two factors called bias and variance which exist when modelling a machine learning algorithm.
- Understand the different challenges in model selection and model evaluation. Study popular re-sampling model selection method called Cross-Validation (K-fold, LOOCYV, etc.) to tune machine learning model.
- Introduce the various learning frameworks and learn to evaluate hypothesis.

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 t, 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.

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 x be the input and χ be the input space, which is the set of all inputs, and Y is the output space, which is the set of all possible outputs, that is, yes/no.

Let D be the input dataset with examples, $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$ for n inputs.

- 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: y — 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.

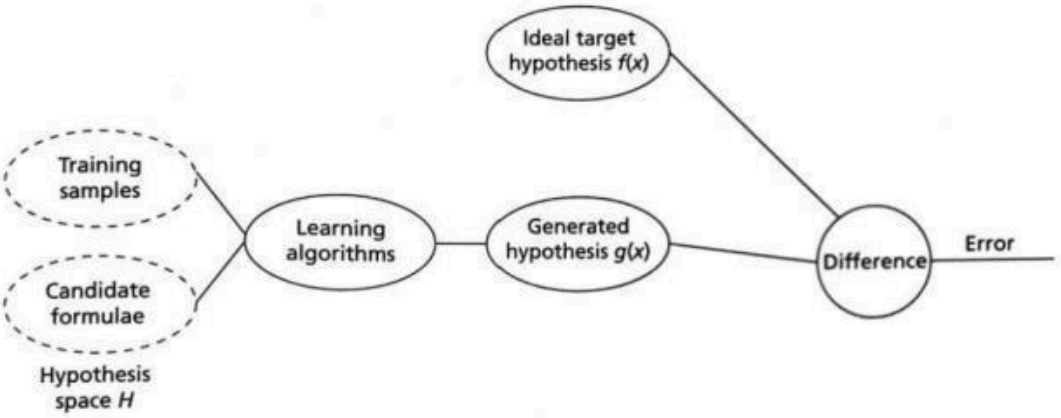

Figure 3.1: Learning Environment

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

Classical and Adaptive Machine Learning Systems

- classical machine learning system has components such as Input, Process and Output. The input values are taken from the environment directly.

- These values are processed and a hypothesis is generated as output model. This model is then used for making predictions.
- The predicted values are consumed by the environment. In contrast to the classical systems, adaptive systems interact with the input for getting labelled data as direct inputs are not available.
- This process is called reinforcement learning. In reinforcement learning, a learning agent interacts with the environment and in return gets feedback.
- Based on the feedback, the learning agent generates input samples for learning, which are used for generating the learning model.
- Such learning agents are not static and change their behavior according to the external signal received from the environment.
- The feedback is known as reward and learning here is the ability of the learning agent adapting to the environment based on the reward. These are the characteristics of an adaptive system.

Learning

Types There are different types of learning. Some of the different learning methods are as follows:

1. **Learn by memorization** or learn by repetition also called as rote learning is done by memorizing without understanding the logic or concept.
2. **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.
3. **Learn by being** taught by an expert or a teacher, generally called as passive learning.
4. **Learning by critical thinking,** also called as deductive learning, deduces new facts q, conclusion from related known facts and information.
5. **Self learning,** also called as reinforcement learning, is a self-directed learning thy normally learns from mistakes punishments and rewards.
6. **Leaming to solve problems** is a type of cognitive learning where learning happen in the mind and is possible by devising a methodology to achieve a goal.
7. **Learning by generalizing explanations,** also called as explanation-based learning (EBJ) is another learning method that exploits domain knowledge from experts to improve the accuracy of learned concepts by supervised learning.

Acquiring general concept from specific instances of the training dataset is the main challenge of machine 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:

1. How can a learning system predict an unseen instance?
2. How do the hypothesis h is close to f, when hypothesis f itself is unknown?
3. How many samples are required?
4. Can we measure the performance of a learning system?
5. Is the solution obtained local or global?

- 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.
- Computational Learning Theory uses many concepts from diverse areas such as Theoretical Computer Science, Artificial Intelligence and Statistics.
- The core concept of COLT is the concept of learning framework. One such important framework is PAC.
- It deals with Probably Approximate Learning (PAC) and Vapnik-Chervonenkis (VC) dimensions.
- The learning framework is discussed in a detailed manner in COLT focuses on supervised learning tasks. Since the complexity of analyzing is difficult, normally, binary classification tasks are considered for analysis.

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:

1. Choosing a training experience
2. Choosing a target function
3. Representation of a target function
4. 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:

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

where, x_1, x_2 and x_3 represent different board features and w_0, w_1, w_2 and w_3 represent weights.

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:

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

- 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.

Thus, the learning system has the following components:

- A Performance system to allow the game to play against itself.
- A Critic system to generate the samples.
- A Generalizer system to generate a hypothesis based on samples.
- An Experimenter system to generate a new system based on the currently learnt function. This is sent as input to the performance system.

INTRODUCTION TO CONCEPT LEARNING

- Concept learning is a learning strategy of acquiring abstract knowledge or inferring a genera 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.
- Each concept or category obtained by learning is a Boolean valued function which takes a true or false value.
- For example, humans can identify different kinds of animals based on common relevant features and categorize all animals based on specific sets of features.

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.

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".

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

- 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.
- For example, (Tail = Short) A (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 'i' is used to represent a candidate hypothesis.

The training dataset given above has 5 training instances with 8 independent attributes and one dependent attribute. Here, the different hypotheses that can be predicted for the target concept are,

	Horns	Tail	Tusks	Paws	Fur	Color	Hooves	Size
h =	<No	?	Yes	?	?	Black	No	Medium>
				(or)				
h =	<No	?	Yes	?	?	Black	No	Big>

The task is to predict the best hypothesis for the target concept (an elephant). The most general hypothesis can allow any value for each of the attribute.

It is represented as:

<?, ?, ?, ?, ?, ?, ?, ?>. This hypothesis indicates that any animal can be an elephant.

The most specific hypothesis will not allow any value for each of the attribute <φ, φ, φ, φ, φ, φ, φ, φ>. This hypothesis indicates that no animal can be an elephant.

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:

(Horns = No) ∧ (Tail = Short) ∧ (Tusks = Yes) ∧ (Paws = ?) ∧ (Fur = ?) ∧ (Color = Black) ∧ (Hooves = No) ∧ (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.

- Thus, concept learning can also be called as Inductive Learning that tries to induce a general function from specific training instances.
- That is why a hypothesis is an approximate target function that best maps the inputs to outputs.

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.
- The set of hypotheses that can be generated by a learning algorithm can be further reduced by specifying a language bias.
- 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.

- Hypothesis ordering is also important wherein the hypotheses are ordered from the most specific one to the most general one in order to restrict searching the hypothesis space exhaustively.

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 spacer or a path from the initial state.

Generalization and Specialization

- In order to understand about how we construct this concept hierarchy, Jet 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.

1. Specialization – General to Specific learning
2. Generalization – Specific to General learning

Generalization - Specific to General Learning This learning methodology will search through the hypothesis space for an approximate hypothesis by generalizing the most specific hypothesis.

Example:- 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.

$h = \langle \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \rangle$

Read the first instance $I1$, to generalize the hypothesis h so that this positive instance can be classified by the hypothesis $h1$.

$I1$:	No	Short	Yes	No	No	Black	No	Big	**Yes (Positive instance)**
$h1$ =	<No	Short	Yes	No	No	Black	No	Big>	

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 This learning methodology will search through the hypothesis space for an approximate hypothesis by specializing the most general hypothesis.

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)**
$h1$ =	<?	?	?	?	?	?	?	?>	

*I*2 :	Yes	Short	No	No	No	Brown	Yes	Medium	**No (Negative instance)**
$h2$ =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	No	?>	
	<?	?	?	?	?	?	?	Big>	

$h2$ 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)**
$h3$ =	<No	?	?	?	?	?	?	?>	
	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	No	?>	
	<?	?	?	?	?	?	?	Big>	
*I*4:	No	Long	No	Yes	Yes	White	No	Medium	**No (Negative instance)**
$h4$ =	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	?	Big>	

Remove any hypothesis inconsistent with this negative instance.

*I*5:	No	Short	Yes	Yes	Yes	Black	No	Big	**Yes (Positive instance)**
$h5$ =	<?	?	Yes	?	?	?	?	?>	
	<?	?	?	?	?	Black	?	?>	
	<?	?	?	?	?	?	?	Big>	

Thus, $h5$ 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. *I*1 is a positive instance, so generalize the most specific hypothesis '*h*' to include this positive instance. Hence,

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

h = <≥9 Yes Excellent Good Fast Yes>

Step 3: Scan the next instance *I*2, since *I*2 is a positive instance. Generalize '*h*' to include positive instance *I*2. 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 *I*2, so put a '?'.

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

h = <≥9 Yes ? Good Fast Yes>

Now,scan *I*3. Since it is a negative instance, ignore it. Hence, the hypothesis remains the same without any change after scanning *I*3.

I3: ≥8 No Good Good Fast No **Negative instance**

h = <≥9 Yes ? Good Fast Yes>

Now scan *I*4. Since it is a positive instance, check for mismatch in the hypothesis '*h*' with *I*4. 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 = <≥9 Yes ? Good ? ?>

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

Limitations of Find-S Algorithm

1. 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.
2. The algorithm finds only one unique hypothesis, wherein there may be many other hypotheses that are consistent with the training dataset.
3. Many times, the training dataset may contain some errors; hence such inconsistent data instances 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.

- This algorithm works fine if the hypothesis space is finite but practically it is difficult to deploy this algorithm. Hence, a variation of this idea is introduced in the Candidate Elimination algorithm.

Version Spaces and the Candidate Elimination Algorithm

Version space learning is to generate all consistent hypotheses around. This algorithm computes the version space by the combination of the two cases namely,

- Specific to General learning – Generalize *S* to include the positive example
- General to Specific learning – Specialize *G* to exclude the negative example

- The algorithm defines two boundaries called 'general boundary' which is a set of all hypotheses that are the most general and "specific boundary' which is a set of all hypotheses that are the most specific.
- Thus, the algorithm limits the version space to contain only those hypotheses that are most general and most specific. Thus, it provides a compact representation of List-then 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**,
     - 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.
     - Prune *G* to exclude all inconsistent hypotheses in *G* with the positive instance.
   - If the instance is **negative,**
     - 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 '?'.
     - Remove from *S* all inconsistent hypotheses with the negative instance.

Generating Positive Hypothesis 'S'

- If it is a positive example, refine S to include the positive instance. We need to generalize S to include the positive instance.
- The hypothesis is the conjunction of 'S' and positive instance. When generalizing, for the first positive instance, add to S all minimal generalizations such that S is filled with attribute values of the positive instance. previous iteration.
- If it is a negative instance, it skips

Generating Negative Hypothesis 'G'

- If it is a negative instance, refine G to exclude the, negative instance.
- Then, prune G to exclude all inconsistent hypotheses in G with the positive, instance. The idea is to add to G al] minimal specializations to exclude the negative instance ap, d be consistent with the positive instance. Negative hypothesis indicates general hypothesis.

Generating Version Space - [Consistent Hypothesis]

We need to take the combination of sets in 'G and check that with 'S'. When the combined set fields are matched with fields in °S', the only that is included in the version space as consistent hypothesis.

Example:-

Consider the same set of instances from the training dataset shown in Table 33 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 G2 to exclude the negative example but stay consistent with S2. Generate hypothesis for each of the non-matching attribute value in S2 and fill with the attribute value of S2.

There is no inconsistent hypothesis in 52 with the negative instance, hence S3 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 G3 is inconsistent with this positive instance, remove the third one. The resulting G4 is,

G4 = <≥9 ? ? ? ? ?>
<? Yes ? ? ? ?>

Using the two boundary sets, S4 and G4, 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.

The diagrammatic representation of deriving the version space is shown in Figure 3.2.

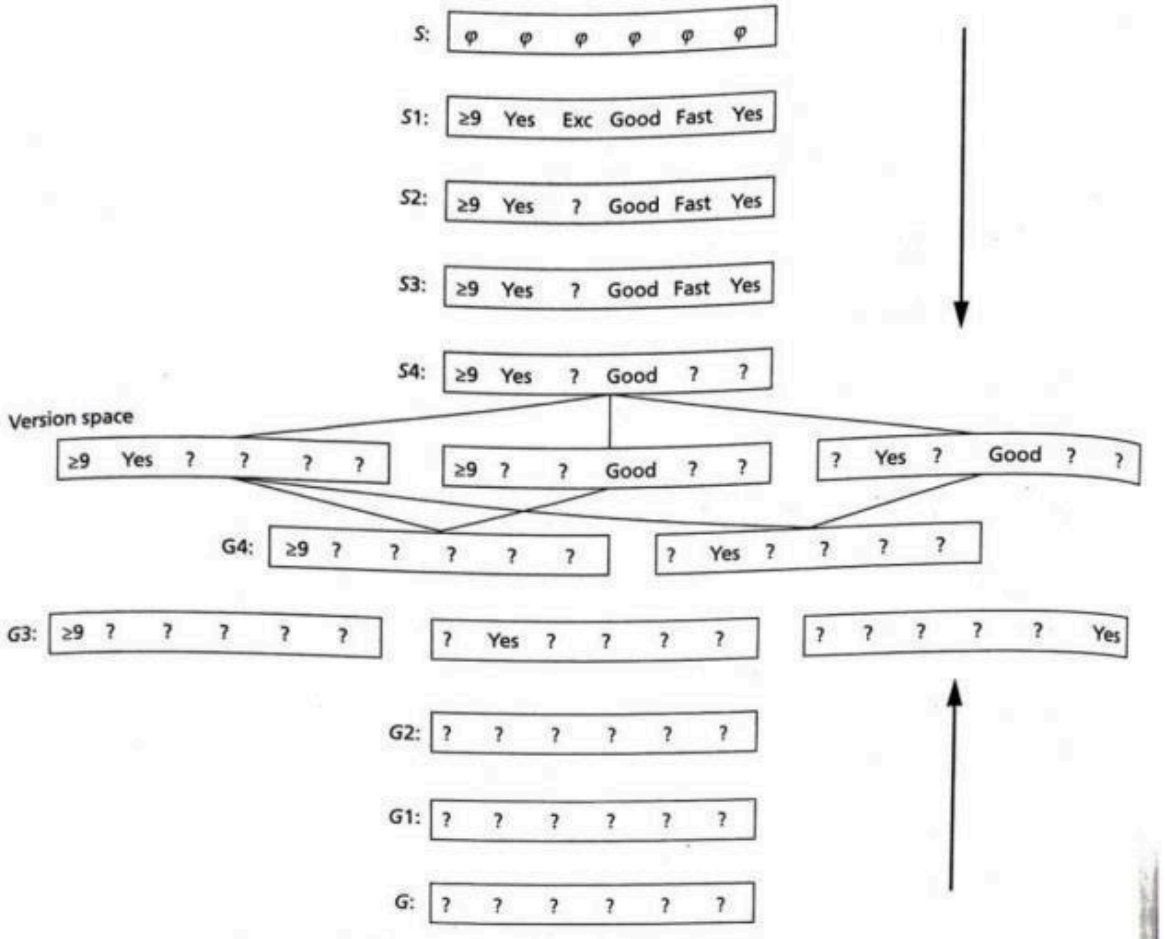

Figure 3.2: Deriving the Version Space

Chapter: - 02 - Similarity-based Learning

- Similarity-based Learning is a supervised learning technique that predicts the class label of a test instance by gauging the similarity of this test instance with training instances.
- Similarity-based learning refers to a family of instance-based learning which is used to solve both classification and regression problems.
- Similarity-based classification is useful in various fields such as image processing, text classification, pattern recognition, bio informatics, data mining, information retrieval, natural language processing, etc.
- A practical application of this learning is predicting daily stock index price changes.

Learning Objectives

- Understand the fundamentals of Instance based learning
- Know about the concepts of Nearest-Neighbor Learning using the algorithm called *k*-Nearest-Neighbors (*k*-NN)
- Learn about Weighted *k*-Nearest-Neighbor classifier that chooses the neighbors by using the weighted distance
- Gain knowledge about Nearest Centroid classifier, a simple alternative to *k*-NN classifiers
- Understand Locally Weighted Regression (LWR) that approximates the linear functions of all *k* neighbors to minimize the error while prediction

INTRODUCTION TO SIMILARITY OR INSTANCE-BASED LEARNING

- 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.
- 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.
- Generally, Similarity-based classification problems formulate the features of test instance and training instances in Euclidean space to learn the similarity or dissimilarity between instances.

Differences Between Instance- and Model-based Learning

- An instance is an entity or an example in the training dataset. It is described by a set of features or attributes. One attribute describes the class label or category of an instance.
- Instance-based methods learn or predict the class label of a test instance only when a new instance is given for classification and until then it delays the processing of the training dataset.

The differences between Instance-based Learning and Model-based Learning

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

Some examples of Instance-based learning algorithms are:

1. k-Nearest Neighbor (*k*-NN)
2. Variants of Nearest Neighbor learning
3. Locally Weighted Regression
4. Learning Vector Quantization (LVQ)
5. Self-Organizing Map (SOM)
6. 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 Q the test instance and classifies it to that category which has the largest probability.
- A visual representation x of this learning is shown in Figure 4.1. There are two classes of objects called C, and C, 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 k-Nearest Neighbor Learning T is predicted as C,.

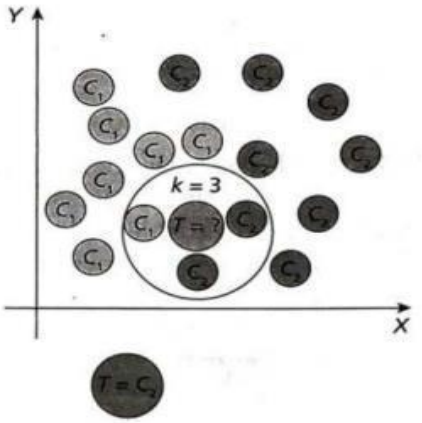

Figure 4.1: Visual Representation of *k*-Nearest Neighbor Learning

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.

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. 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.

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.

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.

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.
- LW is also referred to as a memory-based method as it requires training data while prediction by uses only the training data instances locally around the point of interest.
- Using nearest neighbor g 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 approx.
- imate the linear functions of all '¥ neighbors that minimize the error such that the prediction line is no more linear but rather it is a curve.

Chapter: - 03 – Regression Analysis

- Regression analysis is a supervised learning method for predicting Continuous variables, The difference between classification and regression analysis is that regression methods are used to predict qualitative variables or continuous numbers unlike categorical variables or labels.
- it is used to predict linear or non-linear relationships among variables of the given dataset. This chapter deals with an introduction of regression and its various types.

Learning Objectives

- Understand the basics of regression analysis
- Introduce concepts of correlation and causation
- Learn about linear regression and its validation techniques
- Discuss about multiple linear regression
- Introduce logistic regression
- Study about the concept of regularization
- Study popular regression methods like Ridge, Lasso, and Elastic Net

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.
- 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:

1. What is the relationship between the variables?
2. What is the strength of the relationships?
3. What is the nature of the relationship such as linear or non-linear?
4. What is the relevance of the attributes?
5. What is the contribution of each attribute?

There are many applications of regression analysis. Some of the applications of regressions include predicting:

1. Sales of a goods or services
2. Value of bonds in portfolio management
3. Premium on insurance companies
4. Yield of crops in agriculture
5. 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.
- 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.
- While correlation is about relationships among variables, say ¥ and y, regression is about predicting one variable given another variable.

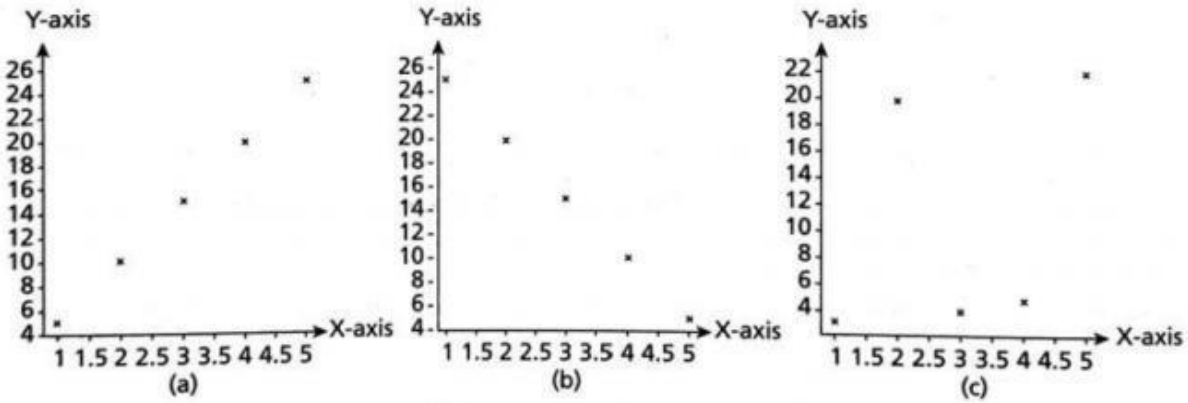

Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation (c) Random Points with No Correlation

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.
- Similarly, the relationship between higher sales of cool drinks due to a rise in temperature is not a causal relation. Even though high temperature is the cause of cool drinks sales, it depends on other factors too,

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 +p 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.

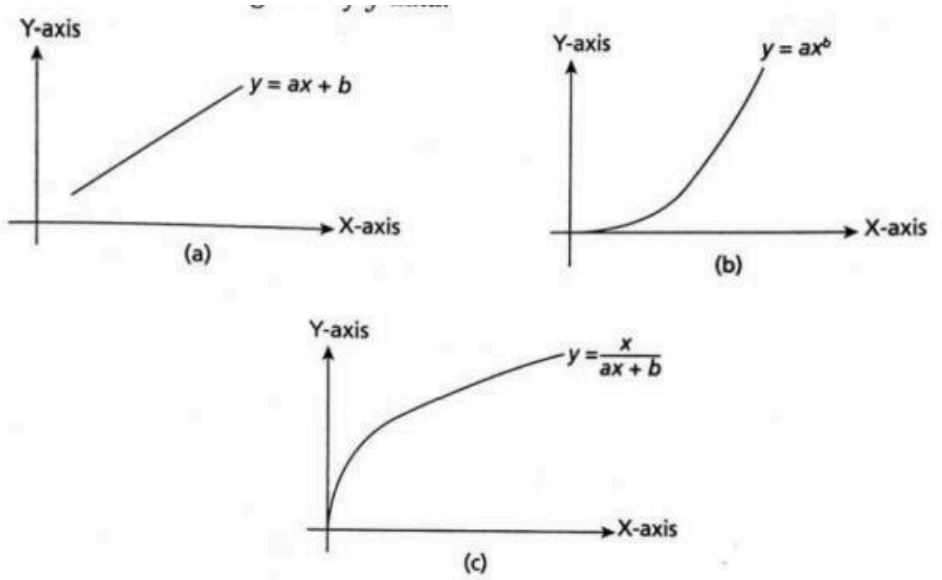

Figure 5.2: (a) Example of Linear Relationship of the Form $y = ax + b$ (b) Example of a Non-linear Relationship of the Form $y = ax^b$ (c) Examples of a Non-linear Relationship $y = \frac{x}{ax + b}$

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

The classification of regression methods is shown in Figure 5.3.

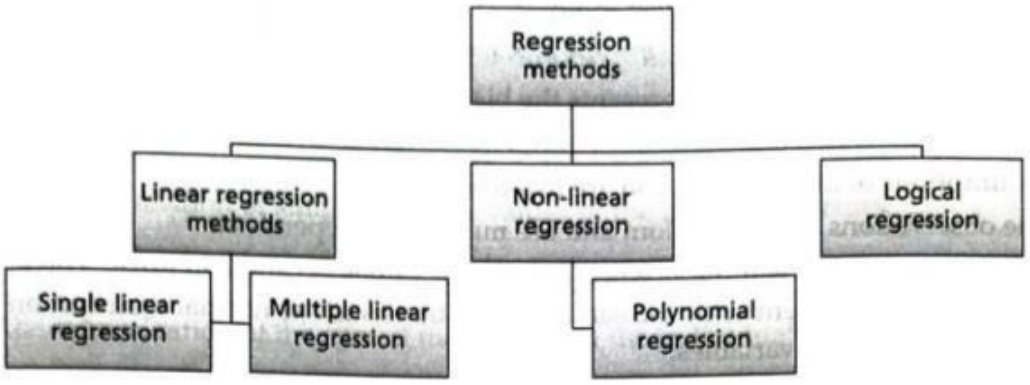

Figure 5.3: Types of Regression Methods

- **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.
- **Polynomial Regression** It is a type of non-linear regression method of describing relationships among variables where N™ 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. At least 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_0 is the intercept which represents the bias and a_1 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:

1. The observations (y) are random and are mutually independent.
2. 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.
3. The distribution of the error term is independent of the joint distribution of explanatory variables.
4. 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.
- 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.

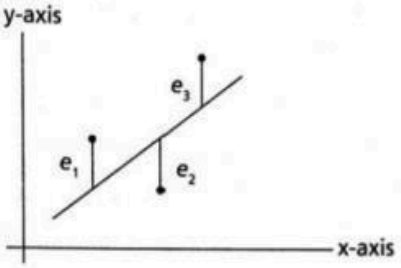

Figure 5.4: Data Points and their Errors

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, ..., x_n) are:

$$y_1 = (a_0 + a_1 x_1) + e_1$$
$$y_2 = (a_0 + a_1 x_2) + e_2$$
$$\vdots$$
$$y_n = (a_0 + a_1 x_n) + e_n \quad (5.3)$$

In general, the error is given as: $e_i = y_i - (a_0 + a_1 x_i)$ (5.4)

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

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, ais a 2 x 1 column vector and e is an n x 1 column vector.
- 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 bias.

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

The regression is given as:

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

The computation order of this equation is shown step by step as:

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

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

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

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

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

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.

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:

$$MAE = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i| \quad (5.12)$$

Here, $\hat{y}$ 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.

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:

$$\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 \quad (5.13)$$

Root Mean Square Error (RMSE)

The square root of the MSE is called RMSE. This is given as:

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

Standard Error

Residuals or error is the difference between the actual (y) and predicted value (7). 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.

Relative MSE

Relative MSE is the ratio of the prediction ability of the $\hat{y}$ 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:

$$\text{RelMSE} = \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2} \quad (5.15)$$

Coefficient of Variation

Coefficient of variation is unit less and is given as:

$$CV = \frac{RMSE}{\bar{y}}$$

Coefficient of Determination

To understand the coefficient of determination, one needs to understand the total variation of coefficients in regression analysis. The sum of the squares of the differences between the y-value of the data pair and the average of y is called total variation. Thus, the following variations can be defined.

The explained variation is given as:

$$= \Sigma(\hat{y}_i - \bar{y}_i)^2 \quad (5.17)$$

The unexplained variation is given as:

$$= \Sigma(y_i - \hat{y}_i)^2 \quad (5.18)$$

Thus, the total variation is equal to the explained variation and the unexplained variation.

The coefficient of determination r^2 is the ratio of the explained and total variations.

$$r^2 = \frac{\text{Explained variation}}{\text{Total variation}} \quad (5.19)$$

Standard Error Estimate

Standard error estimate is another useful measure of regression. It is the standard deviation of the observed values to the predicted values. This is given as:

$$s_e = \sqrt{\frac{\Sigma(y_i - \hat{y}_i)}{n-2}} \quad (5.20)$$

Here, as usual, y_i is the observed value and $\hat{y}_i$ is the predicted value. Here, n is the number of samples.

Example: -

Let us consider the data given in the Table 5.3 with actual and predicted values. Find standard error estimate.

Solution: The observed value or the predicted value is given below in Table 5.6.

Table 5.6: Sample Data

x_i	y_i	Predicted Value	$(y - \hat{y})^2$
1	1.5	1.46	$(1.5 - 1.46)^2 = 0.0016$
2	2.9	2.02	$(2.9 - 2.02)^2 = 0.7744$
3	2.7	2.58	$(2.7 - 2.58)^2 = 0.0144$
4	3.1	3.14	$(3.1 - 3.14)^2 = 0.0016$

The sum of $(y - \hat{y})^2$ for all $i = 1, 2, 3$ and 4 (i.e., number of samples $n = 4$) is 0.792. The standard deviation error estimate as given in Eq. (5.20) is:

$$\sqrt{\frac{0.792}{4-2}} = \sqrt{0.396} = 0.629$$