MODULE 3:

CHAPTER 3

3. Basics of Learning Theory

3.1 Introduction to Learning and its Types

3.2 Introduction to Computation Learning Theory

3.3 Design of a Learning System

3.4 Introduction to Concept Learning

- 3.4.1 Representation of a Hypothesis
- 3.4.2 Hypothesis Space
- 3.4.3 Heuristic Space Search
- 3.4.4 Generalization and Specialization
- 3.4.5 Hypothesis Space Search by Find-S Algorithm
- 3.4.6 Version Spaces

CHAPTER 4

4. Similarity-based Learning

4.1 Introduction to Similarity or Instance-based Learning

- 4.1.1 Differences Between Instance and Model-based Learning

4.2 Nearest-Neighbor Learning

4.3 Weighted K-Nearest-Neighbor Algorithm

4.4 Nearest Centroid Classifier

4.5 Locally Weighted Regression (LWR)

CHAPTER 5

5. Regression Analysis

5.1 Introduction to Regression

5.2 Introduction to Linearity, Correlation, and Causation

5.3 Introduction to Linear Regression

5.4 Validation of Regression Methods

CHAPTER 3

BASICS OF LEARNING THEORY

INTRODUCTION TO LEARNING AND ITS TYPES

- Learning is a process by which one can acquire knowledge and construct new ideas or concepts based on the experiences.
- 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.
- 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)

- Let x- input, χ-input space, Y –is the output space. Which is the set of all possible outputs, that is yes/no,
- Let D –dataset for n inputs. Consider, target function be: χ-> Y , that maps input to output.
- **Objective**: To pick a function, g: χ-> Y to appropriate hypothesis f.

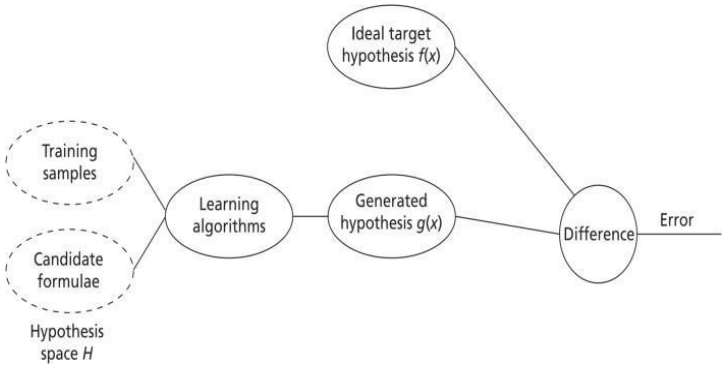


Fig: Learning Environment

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:

$\sum_{i=1}^{D} x_i w_i > Threshold$, belongs to class 1 and

$\sum_{i=1}^{D} x_i w_i < Threshold$, belongs to another class

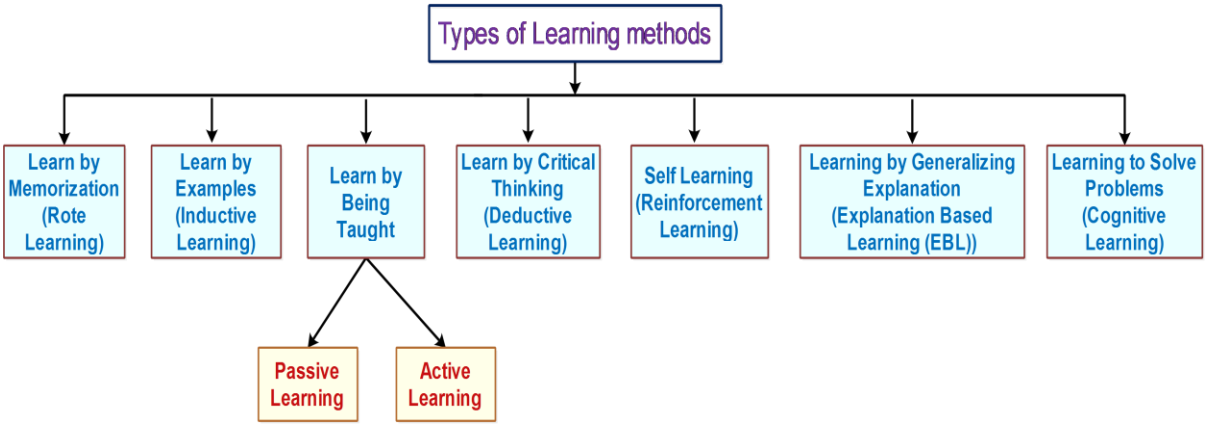
- This can be put into a single equation as follows

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

- **This is called perception learning algorithm.**

Classical and Adaptive ML systems. 3

- **Classic** machines examine data inputs according to a predetermined set of rules, finding patterns and relationships that can be used to generate predictions or choices. Support vector machines, decision trees, and logistic regression are some of the most used classical machine- learning techniques.
- A class of machine learning techniques called **adaptive** machines, commonly referred to as adaptive or deep learning, is created to automatically learn from data inputs without being explicitly programmed.
- By learning hierarchical representations of the input, these algorithms are able to handle more complex and unstructured data, such as photos, videos, and natural language.
- Adaptive ML is the next generation of traditional ML – the new, the improved, the better. Even though traditional ML witnessed significant progress.



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. However, there is a special kind of learning called active learning where the learner car interactively query a teacher/expert to label unlabelled data instances with the desired outputs.

4. Learning by critical thinking, also called as deductive learning, deduces new facts or conclusion from related known facts and information.

5. Self-learning, also called as reinforcement learning, is a self-directed learning that normally learns from mistakes punishments and rewards.

6. 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.

7. 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:

  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).
- Computational Learning Theory (COLT) is like a guide for understanding how computers learn.
- It deals with questions such as how computers predict new things, measure their learning performance, and handle unknown information.
- COLT has two key parts: Probably Approximate Learning (PAC), which looks at how hard learning tasks are, and Vapnik-Chervonenkis (VC) dimensions, which focus on computational capacity.
- In simpler terms, COLT helps us figure out how computers learn by breaking down the process and measuring their abilities, using concepts from computer science, artificial intelligence, and statistics.

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:

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

1. Choosing a 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.

2. 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.

3. 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:

  $V = w0 + w1x1 + w2x2 + w3x3$
- Where, x1, x2 & and x3 represent different board features and w0,w1, w2, and w3 represent weights.

4. 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:

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

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 requires three things:

1. 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.
2. 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.
3. Test – New instances to test the learned model.

3.1.1 Representation of a Hypothesis

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

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.

- For Example, (Tail = Short)^(Color=Black)……
- The set of hypothesis in the search space is called as hypothesis. Hypotheses are the plural form of hypothesis.
- Generally 'H' is used to represent the hypothesis and 'h' is used to represent a candidate hypothesis.
- "?" denotes that the attribute can take any value
- Null indicates that the attribute cannot take any value
- Single value denotes a specific single value from acceptable values of the attribute

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 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.

3.1.2 Hypothesis Space

- *Hypothesis space* is the set of all possible hypotheses that approximates the target function *f*.
- The subset of hypothesis space that is consistent with all-observed training instances is called as **Version Space.**

3.1.3 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
- 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

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.

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 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 $I1$, 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 $I2$, it is a negative instance, so ignore it.

$I2$:	Yes	Short	No	No	No	Brown	Yes	Medium	**No (Negative instance)**
$h2$ =	<No	Short	Yes	No	No	Black	No	Big>	

Similarly, when reading the third instance $I3$, it is a positive instance so generalize $h2$ to $h3$ to accommodate it. The resulting $h3$ is generalized.

$I3$:	No	Short	Yes	No	No	Black	No	Medium	**Yes (Positive instance)**
$h3$ =	<No	Short	Yes	No	No	Black	No	?>	

Ignore $I4$ since it is a negative instance.

$I4$:	No	Long	No	Yes	Yes	White	No	Medium	**No (Negative instance)**
$h4$ =	<No	Short	Yes	No	No	Black	No	?>	

When reading the fifth instance $I5$, $h4$ is further generalized to $h5$.

$I5$:	No	Short	Yes	Yes	Yes	Black	No	Big	**Yes (Positive instance)**
$h5$ =	<No	Short	Yes	?	?	Black	No	?>	

Now, after observing all the positive instances, an approximate hypothesis $h5$ 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.

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	?	?	?	?	?>	

<?	?	?	?	?	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.
   $h = <\varphi \quad \varphi \quad \varphi \quad \varphi \quad \varphi \quad \ldots..>$
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.

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.

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 *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 = <	≥9	Yes	?	Good	?	?>

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

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.
- 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.

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.

Candidate Elimination 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.
- For the subsequent positive instances scanned, check the attribute value of the positive instance and S obtained in the previous iteration.
- If the attribute values of positive instance and S are different, fill that field value with a '??. If the attribute values of positive instance and S are same, no change is required. 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 all minimal specializations to exclude the negative instance and be consistent with the positive instance. Negative hypothesis indicates general hypothesis.

- If the attribute values of positive and negative instances are different, then fill that field with positive instance value so that the hypothesis does not classify that negative instance as true. If the attribute values of positive and negative instances are same, then no need to update 'G' and fill that attribute value with a '?'.

Generating Version Space -

- 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', ther only that is included in the version space as consistent hypothesis.

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 I2. Since I2 is a positive instance, generalize 'S1' to include positive instance I2. For each of the non-matching attribute value in 'S1', put a '?' to include this positive instance. The third attribute value is mismatching in 'S1' with I2, so put a '?'.

I2:	≥9	Yes	Good	Good	Fast	Yes	**Positive instance**
S2 = <	≥9	Yes	?	Good	Fast	Yes>	

Prune G1 to exclude all inconsistent hypotheses with the positive instance. Since G1 is consistent with this positive instance, there is no change. The resulting G2 is,

G2 = <? ? ? ? ? ?>

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.

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

The diagrammatic representation of deriving the version space is shown below:

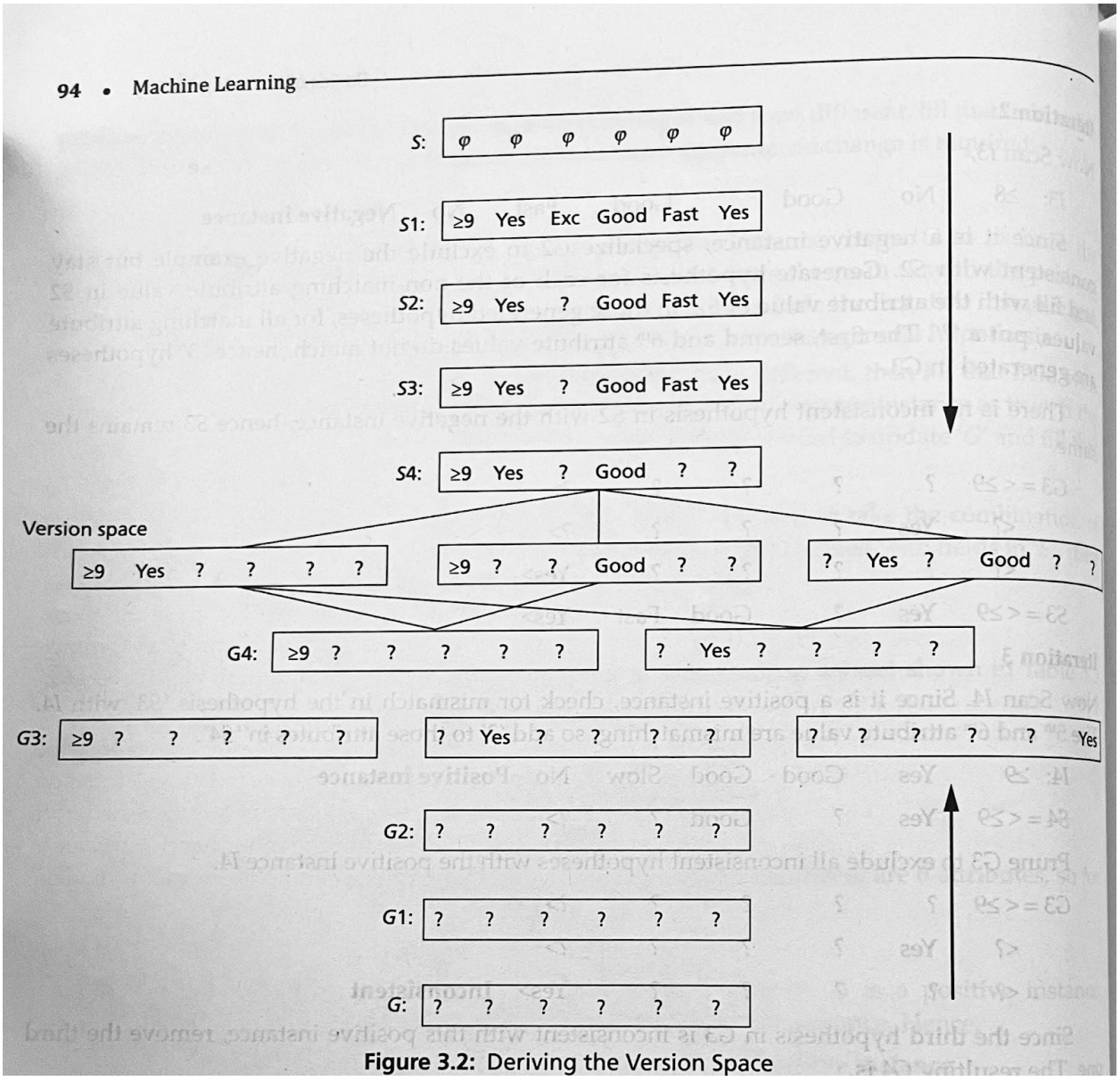


Figure 3.2: Deriving the Version Space

CHAPTER 4

SIMILARITY-BASED LEARNING

Similarity or Instance-based Learning

- It's 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 learning is a branch of machine learning that focuses on training models to recognize the similarity or dissimilarity between data points.
- Its also called instance based learning/just in time learning
- This learning mechanism simply stores all data and uses it only when it needs to classify an unseen instance
- Advantage: processing occurs only when a request to classify an unseen instance is given.
- Drawback: it requires large memory to store the data since a model is not constructed initially with the training data
- Several distance metrics are used to estimate the similarity or dissimilarity between instances required for clustering, nearest neighbor classification etc.
- Popular distance metrics used are hamming distance, Euclidean distance, manhattan distance etc.

Table 4.1: 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

Instance-based Learning	Model-based Learning
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

4.1.1 Difference between Instance-and Model-based Learning

Some examples of Instance-based Learning **algorithms** are:

a) KNN
b) Variants of KNN
c) Locally weighted regression
d) Learning vector quantization
e) Self-organizing maps
f) RBF networks

Nearest-Neighbor Learning

- A powerful classification algorithm used in pattern recognition.
- K nearest neighbors stores all available cases and classifies new cases based on a similarity measure (e.g distance function)
- One of the top data mining algorithms used today.
- A non-parametric lazy learning algorithm (An Instance based Learning method).

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.

- Used for both classification and regression problem:

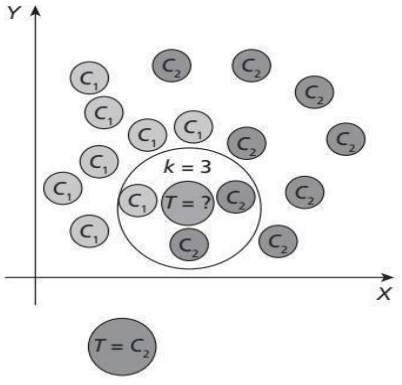
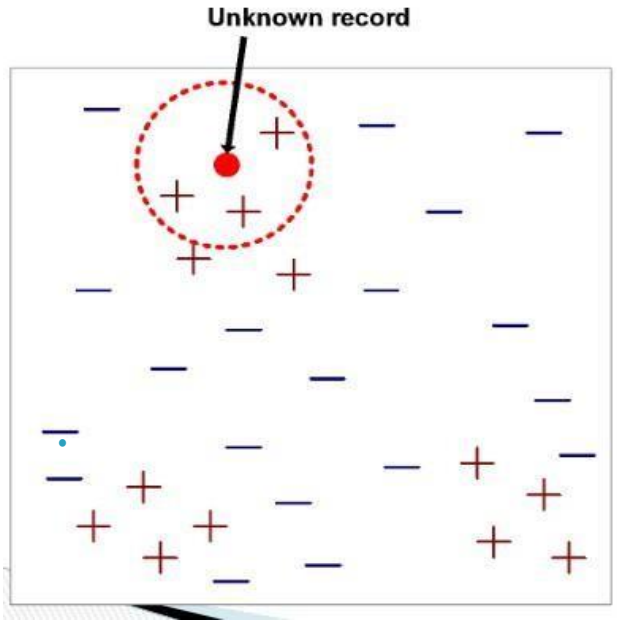


Figure 4.1: Visual Representation of *k*-Nearest Neighbor Learning

Here, 2 classes of objects called C_1 and C_2. 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 C_2.



- Requires three thing
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of k, the number of nearest neighbors to retrieve

- To classify an unknown record:
  1. Compute distance to other training records
  2. Identify k nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

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

(Continued)

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'.

- Data normalization/standardization is required when data (features) have different ranges or a wider range of possible values when computing distances and to transform all features to a specific range.
- This is probably done to eliminate the influence of one feature over another (i.e., to give all features equal chances).

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 influe

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.

Problem

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.

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: 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

The Nearest Centroids algorithm assumes that the centroids in the input feature space are different for each target label. The training data is split into groups by class label, then the centroid for each group of data is calculated. Each centroid is simply the mean value of each of the input variables, so it is also called as Mean Difference classifier. If there are two classes, then two centroids or points are calculated; three classes give three centroids, and so on.

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: Consider the sample data shown in Table 4.9 with two features r 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'.

124 • Machine Learning

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'.

4.2 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.
- Ordinary linear regression finds out a linear relationship between the input x and the output y.
- Given training dataset I,

Hypothesis function $h_\beta(x)$, the predicted target output is a linear function where β_0 is the intercept and β_1 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 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]

Weight of Instance 4 is:

$$w_4 = e^{\frac{-(x_4-x)^2}{2\tau^2}} = e^{\frac{-(1-2)^2}{2\times 0.4^2}} = e^{-3.125} = 0.043$$

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 β parameters.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

CHAPTER 5
REGRESSION ANALYSIS

5.1 Introduction to Regression

Regression analysis is a fundamental concept that consists of a set of machine learning methods that predict a continuous outcome variable (y) based on the value of one or multiple predictor variables (x).

OR

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.

Regression is a supervised learning technique which helps in finding the correlation between variables.

It is mainly used for prediction, forecasting, time series modelling, and determining the causal-effect relationship between variables**.**

Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum." The distance between datapoints and line tells whether a model has captured a strong relationship or not.

- Function of regression analysis is given by:

  Y=f(x)

Here, y is called dependent variable and x is called independent variable.

Applications of Regression Analysis

- Sales of a goods or services
- Value of bonds in portfolio management
- Premium on insurance componies
- Yield of crop in agriculture
- Prices of real estate

5.2 INTRODUCTION TO LINEARITY, CORRELATION AND CAUSATION

A correlation is the statistical summary of the relationship between two sets of variables. It is a core part of data exploratory analysis, and is a critical aspect of numerous advanced machine learning techniques.

Correlation between two variables can be found using a **scatter** plot

There are different types of correlation:

Positive Correlation: Two variables are said to be positively correlated when their values move in the same direction. For example, in the image below, as the value for X increases, so does the value for Y at a constant rate.

Negative Correlation: Finally, variables X and Y will be negatively correlated when their values change in opposite directions, so here as the value for X increases, the value for Y decreases at a constant rate.

Neutral Correlation: No relationship in the change of variables X and Y. In this case, the values are completely random and do not show any sign of correlation, as shown in the following image:

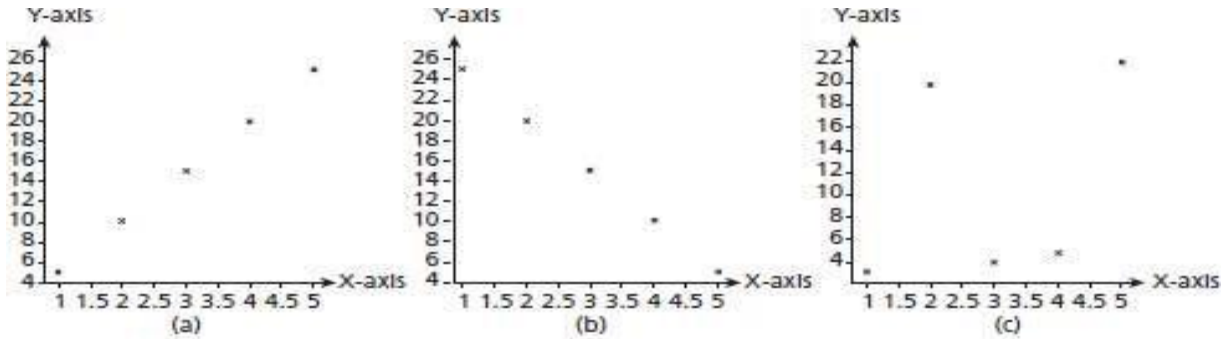


Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation (c) Random Points with No Correlation

Causation

Causation is about relationship between two variables as x causes y. This is called x implies b. Regression is different from causation. Causation indicates that one event is the result of the occurrence of the other event; i.e. there is a causal relationship between the two events.

Linear and Non-Linear Relationships

The relationship between input features (variables) and the output (target) variable is fundamental. These concepts have significant implications for the choice of algorithms, model complexity, and predictive performance.

Linear relationship creates a straight line when plotted on a graph, a Non-Linear relationship does not create a straight line but instead creates a curve.

Example:

Linear-the relationship between the hours spent studying and the grades obtained in a class.

Non-Linear-

Linearity:

Linear Relationship: A linear relationship between variables means that a change in one variable is associated with a proportional change in another variable. Mathematically, it can be represented as y = a * x + b, where y is the output, x is the input, and a and b are constants.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

Linear Models: Goal is to find the best-fitting line (plane in higher dimensions) to the data points. Linear models are interpretable and work well when the relationship between variables is close to being linear.

Limitations: Linear models may perform poorly when the relationship between variables is non-linear. In such cases, they may underfit the data, meaning they are too simple to capture the underlying patterns.

Non-Linearity:

Non-Linear Relationship: A non-linear relationship implies that the change in one variable is not proportional to the change in another variable. Non-linear relationships can take various forms, such as quadratic, exponential, logarithmic, or arbitrary shapes.

Non-Linear Models: Machine learning models like decision trees, random forests, support vector machines with non-linear kernels, and neural networks can capture non-linear relationships. These models are more flexible and can fit complex data patterns.

Benefits: Non-linear models can perform well when the underlying relationships in the data are complex or when interactions between variables are non-linear. They have the capacity to capture intricate patterns.

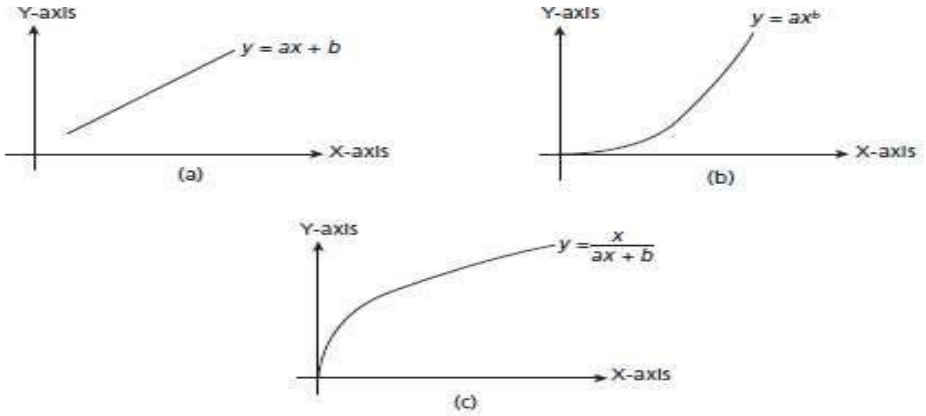


Figure 5.2: (a) Example of Linear Relationship of the Form $y = ax + b$ (b) Example of a Non-linear Relationship of the Form $y = ax^b$ (c) Examples of a Non-linear Relationship $y = \frac{x}{ax + b}$

Types of Regression

Types of Regression Methods

The classification of regression methods is shown in Figure 5.3.

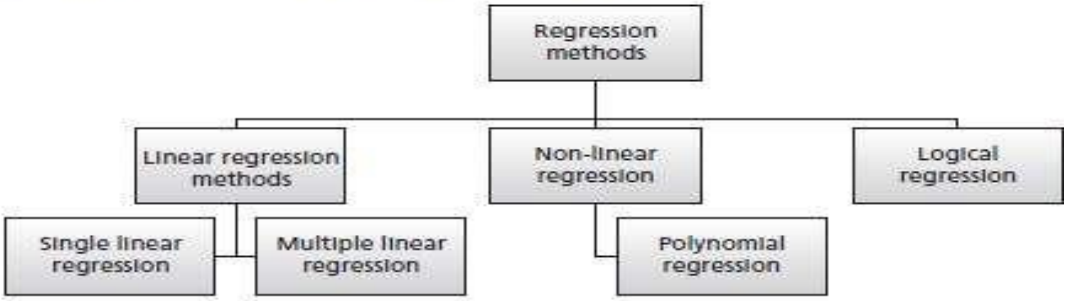


Figure 5.3: Types of Regression Methods

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

Linear Regression:

Single Independent Variable: Linear regression, also known as simple linear regression, is used when there is a single independent variable (predictor) and one dependent variable (target).

Equation: The linear regression equation takes the form: $Y = \beta0 + \beta1X + \varepsilon$, where Y is the dependent variable, X is the independent variable, β0 is the intercept, β1 is the slope (coefficient), and ε is the error term.

Purpose: Linear regression is used to establish a linear relationship between two variables and make predictions based on this relationship. It's suitable for simple scenarios where there's only one predictor.

Multiple Regression:

Multiple Independent Variables: Multiple regression, as the name suggests, is used when there are two or more independent variables (predictors) and one dependent variable (target).

Equation: The multiple regression equation extends the concept to multiple predictors: $Y = \beta0 + \beta1X1 + \beta2X2 + ... + \betanXn + \varepsilon$, where Y is the dependent variable, X1, X2, ..., Xn are the independent variables, β0 is the intercept, β1, β2, ..., βn are the coefficients, and ε is the error term.

Purpose: Multiple regression allows you to model the relationship between the dependent variable and multiple predictors simultaneously. It's used when there are multiple factors that may influence the target variable, and you want to understand their combined effect and make predictions based on all these factors.

Polynomial Regression:

Use: Polynomial regression is an extension of multiple regression used when the relationship between the independent and dependent variables is non-linear.

Equation: The polynomial regression equation allows for higher-order terms, such as quadratic or cubic terms: $Y = \beta0 + \beta1X + \beta2X^2 + ... + \betanX^n + \varepsilon$. This allows the model to fit a curve rather than a straight line.

Logistic Regression:

Use: Logistic regression is used when the dependent variable is binary (0 or 1). It models the probability of the dependent variable belonging to a particular class.

Equation: Logistic regression uses the logistic function (sigmoid function) to model probabilities: $P(Y=1) = 1 / (1 + e^{(-z)})$, where z is a linear combination of the independent variables: $z = \beta0 + \beta1X1 + \beta2X2 + ... + \betanXn$. It transforms this probability into a binary outcome.

Lasso Regression (L1 Regularization):

Use: Lasso regression is used for feature selection and regularization. It penalizes the absolute values of the coefficients, which encourages sparsity in the model.

Objective Function: Lasso regression adds an L1 penalty to the linear regression loss function: Lasso = RSS + λΣ|βi|, where RSS is the residual sum of squares, λ is the regularization strength, and |βi| represents the absolute values of the coefficients.

Ridge Regression (L2 Regularization):

Use: Ridge regression is used for regularization to prevent overfitting in multiple regression. It penalizes the square of the coefficients.

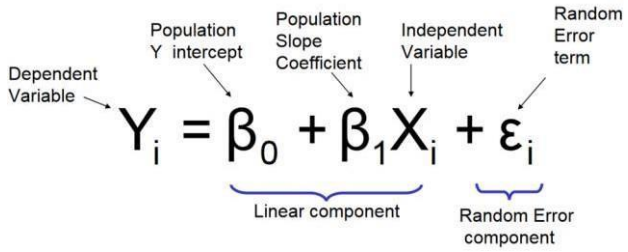
Objective Function: Ridge regression adds an L2 penalty to the linear regression loss function: Ridge = RSS + λΣ(βi^2), where RSS is the residual sum of squares, λ is the regularization strength, and (βi^2) represents the square of the coefficients.

Limitations of Regression

1. Outliers – Outliers are abnormal data. It can bias the outcome of the regression model, as outliers push the regression line towards it.
2. 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.
3. Missing data – Missing data in training data can make the model unfit for the sampled data.
4. 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.

5.3 INTRODUCTION TO LINEAR REGRESSION

Linear regression model can be created by fitting a line among the scattered data points. The line is of the form:



ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

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.

Ordinary Least Square Approach

The ordinary least squares (OLS) algorithm is a method for estimating the parameters of a linear regression model. **Aim**: To find the values of the linear regression model's parameters (i.e., the coefficients) that minimize the sum of the squared residuals.

In mathematical terms, this can be written as: **Minimize** $\sum(\mathbf{y_i} - \hat{\mathbf{y}}_i)^{\wedge}\mathbf{2}$

where **y**i is the actual value, $\hat{\mathbf{y}}$i is the predicted value.
A linear regression model used for determining the value of the response variable, $\hat{\mathbf{y}}$**,** can be represented as the following equation.
y = b0 + b1x1 + b2x2 + … + bnxn + e

- where: y - is the dependent variable, b0 is the intercept, e is the error term
- b1, b2, …, bn are the coefficients of the independent variables x1, x2, …, xn

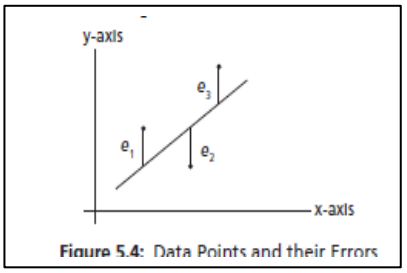


Figure 5.4: Data Points and their Errors

The coefficients b1, b2, …, bn can also be called the **coefficients of determination**. The goal of the OLS method can be used to estimate the unknown parameters (**b1, b2, …, bn**) by minimizing the sum of squared residuals (RSS). The sum of squared residuals is also termed the sum of squared error (SSE).
This method is also known as the **least-squares method** for regression or linear regression.
Mathematically the line of equations for points are:

$y_1=(a_0+a_1x_1)+e_1$

$y_2=(a_0+a_1x_2)+e_2$ and so on

..... $y_n=(a_0+a_1x_n)+e_n$.

In general $\mathbf{e_i=y_i - (a_0+a_1x_1)}$

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

Here, the terms $(e_1, e_2, \ldots, e_n)$ 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 a_0 and a_1 such that:

$$E = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i)) \quad (5.5)$$

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

$$E = \sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} |(y_i - (a_0 + a_1 x_i))| \quad (5.6)$$

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

$$E = \sum_{i=1}^{n} (e_i)^2 = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i))^2 \quad (5.7)$$

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:

$$J(a_1, a_0) = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i)]^2 \quad (5.8)$$

Here, $J(a_0, a_1)$ is the criterion function of parameters a_0 and a_1. This needs to be minimized. This is done by differentiating and substituting to zero. This yields the coefficient values of a_0 and a_1. The values of estimates of a_0 and a_1 are given as follows:

$$a_1 = \frac{(\overline{xy}) - (\overline{x})(\overline{y})}{(\overline{x_i^2}) - (\overline{x})^2} \quad (5.9)$$

And the value of a_0 is given as follows:

$$a_0 = (\overline{y}) - a_1 \times \overline{x} \quad (5.10)$$

Let us consider a simple problem to illustrate the usage of the above concept.

Linear Regression 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

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

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) $= \overline{x} = \frac{15}{5}$ $= 3$	Sum = 12.6 Average of (y_i) $= \overline{y} = \frac{12.6}{5}$ $= 2.52$	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:

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

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

The fitted line is shown in Figure 5.5.

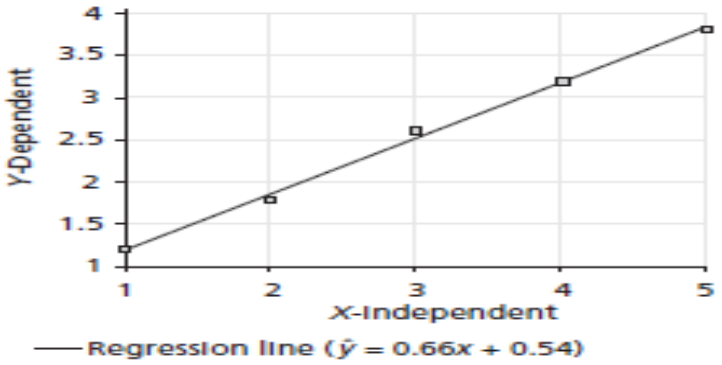


Figure 5.5: Linear Regression Model Constructed

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.

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

Linear Regression in Matrix Form

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:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \quad (5.11)$$

This can be written as:

$Y = Xa + e$, where X is an $n \times 2$ matrix, Y is an $n \times 1$ vector, a is a 2×1 column vector and e is an $n \times 1$ column vector.

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 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 $2.2x - 1.5$.

5.4 VALIDATION OF REGRESSION METHODS

The regression 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 ($\hat{y}$).

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.

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:

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

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING (21CS54)

Root Mean Square Error (RMSE)

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

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2} \tag{5.14}$$

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:

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

Coefficient of Variation

Coefficient of variation is unit less and is given as:

$$\text{CV} = \frac{\text{RMSE}}{\overline{y}} \tag{5.16}$$

Coefficient of Determination

The coefficient of determination (R^2 or r-squared) is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that can be explained by the independent variable.

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 variation can be defined as,

The explained variation is given by, $=\sum(\hat{Y}_i - \text{mean}(Y_i))^2$

The unexplained variation is given by, $=\sum(Y_i - \hat{Y}_i)^2$

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

The coefficient of determination r2 is the ratio of the explained and unexplained variations.

$$r^2 = \frac{\text{Explained variation}}{\text{Total variation}}$$

Consider the following training set Table 5.4 for predicting the sales of the items.

Table 5.4: Training Item Table

Items x_i	Actual Sales (In Thousands) y_i
I_1	80
I_2	90
I_3	100
I_4	110
I_5	120

Consider two fresh items I_6 and I_7, whose actual values are 80 and 75, respectively. A regression model predicts the values of the items I_6 and I_7 as 75 and 85, respectively. Find MAE, MSE, RMSE, RelMSE and CV.

Solution: The test items' actual and prediction is given in Table 5.5 as:

Table 5.5: Test Item Table

Test Items	Actual Value y_i	Predicted Value $\bar{y}_i$
I_6	80	75
I_7	75	85

Mean Absolute Error (MAE) using Eq. (5.12) is given as:

$$MAE = \frac{1}{2} \times |80 - 75| + |75 - 85| = \frac{15}{2} = 7.5$$

Mean Squared Error (MSE) using Eq. (5.13) is given as:

$$MSE = \frac{1}{2} \times |80 - 75|^2 + |75 - 85|^2 = \frac{125}{2} = 62.5$$

Root Mean Square error using Eq. (5.14) is given as:

$$RMSE = \sqrt{MSE} = \sqrt{62.5} = 7.91$$

For finding RelMSE and CV, the training table should be used to find the average of y.

The average of y is $\frac{80 + 90 + 100 + 110 + 120}{5} = \frac{500}{5} = 100.$

RelMSE using Eq. (5.15) can be computed as:

$$RelMSE = \frac{(80 - 75)^2 + (75 - 85)^2}{(80 - 100)^2 + (75 - 100)^2} = \frac{125}{1025} = 0.1219$$

CV can be computed using Eq. (5.16) as $\frac{\sqrt{62.5}}{100} = 0.08.$

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 5.4: 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:

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

THANK YOU