MODULE 3

BASICS OF LEARNING THEORY

3.1 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.

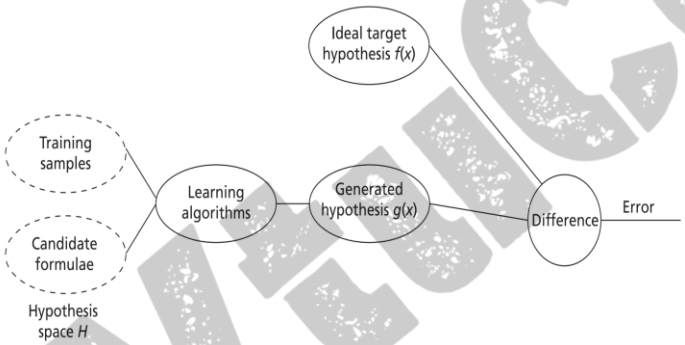


Fig: Learning Environment

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:

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

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

This can be put into a single equation as follows:

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

where, $x_1, x_2, \cdots, x_d$ are the components of the input vector, $w_1, w_2, \ldots, w_d$ are the weights and +1 and −1 represent the class. This simple model is called perception model. One can simplify this by making $w_0 = b$ and fixing it as 1, then the model can further be simplified as:

$$h(x) = sign(w^T x).$$

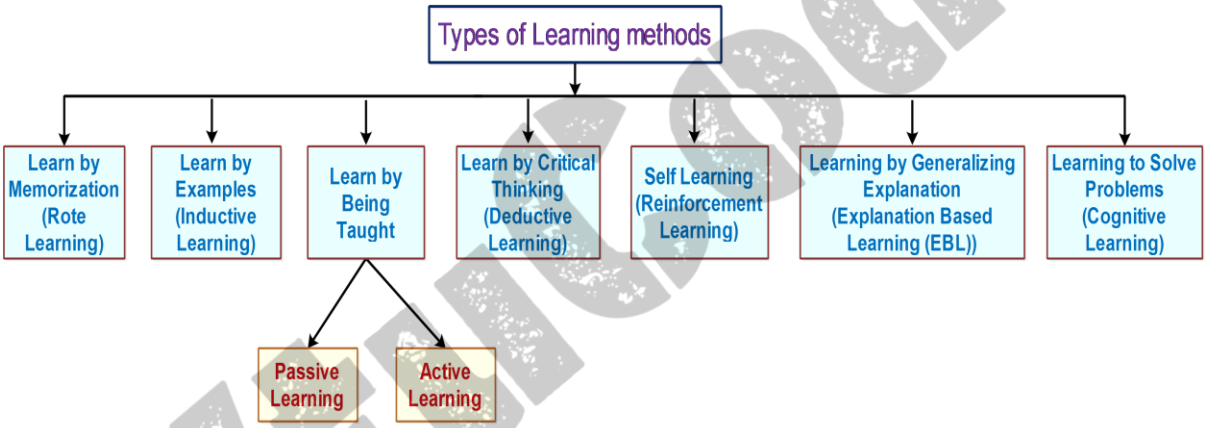
Classical and Adaptive ML systems.

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



3.2 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).

3.3 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

3.4 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.

3.4.1 Representation of a Hypothesis

A *hypothesis* '*h*' approximates a target function '*f*' to represent the relationship between the independent attributes and the dependent attribute of the training instances. The hypothesis is the predicted approximate model that best maps the inputs to outputs. Each hypothesis is represented as a conjunction of attribute conditions in the antecedent part.

3.4.2 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.**

3.4.3 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.

3.4.4 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.

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

3.4.5 Hypothesis Space Search by Find-S Algorithm

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.

3.4.6 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

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.

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

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

Deriving the Version Space

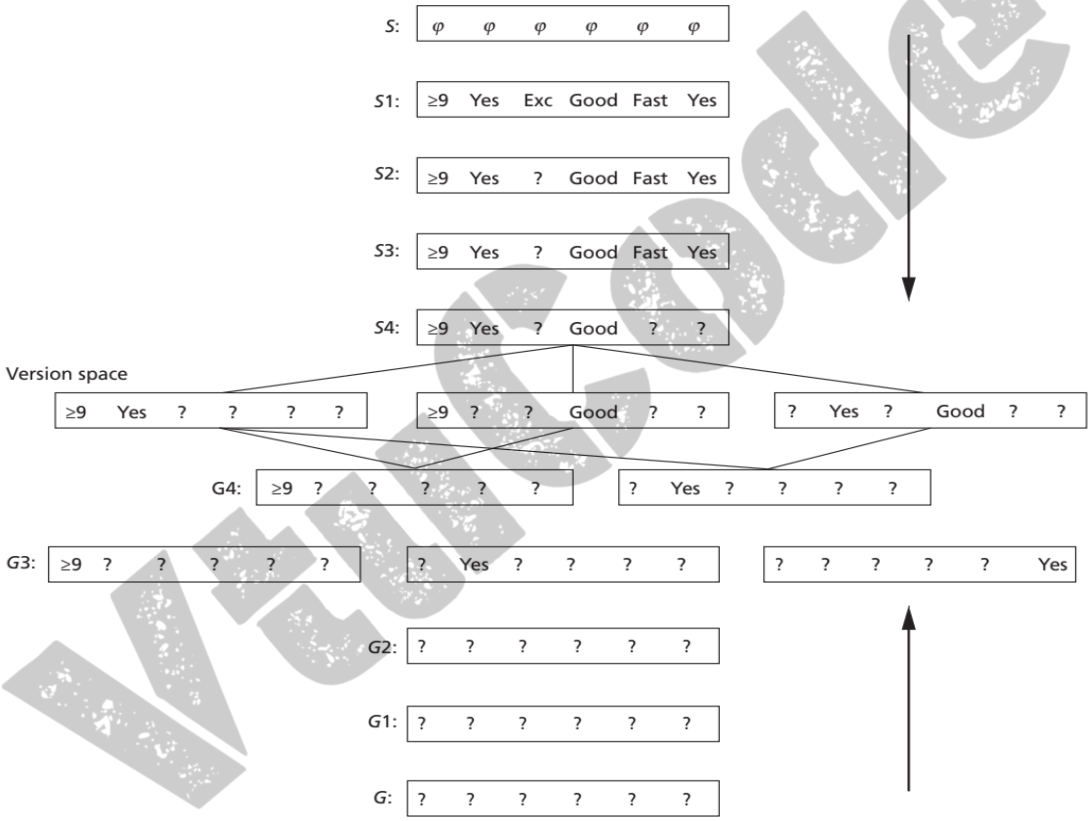


Figure 3.2: Deriving the Version Space

MODULE 3

CHAPTER 4

SIMILARITY-BASED LEARNING

4.1 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. This learning mechanism simply stores all data and uses it only when it needs to classify an unseen instance. The advantage of using this learning is that processing occurs only when a request to classify a new instance is given. This methodology is particularly useful when the whole dataset is not available in the beginning but collected in an incremental manner.

4.1.1 Difference between Instance-and Model-based Learning

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

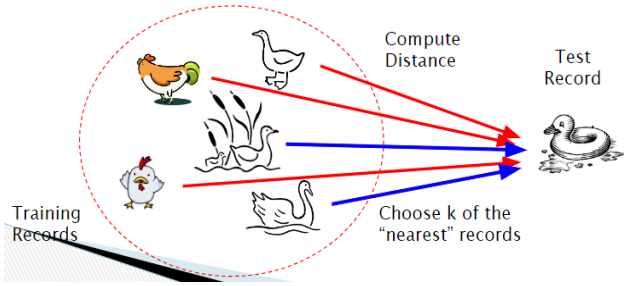
Some examples of Instance-based Learning **algorithms** are:

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

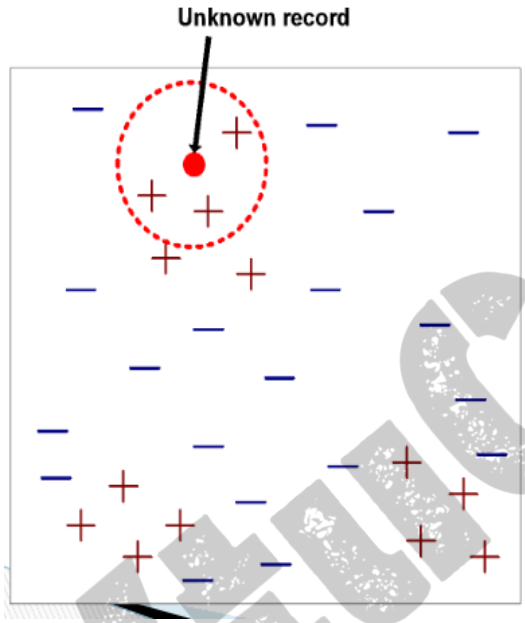
Nearest-Neighbor Learning

- 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).

- Used for both classification and regression problems.
  - Basic idea:
    - "*If it walks like a duck, quacks like a duck, then it's probably a duck*"



- Requires three things
  - 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)



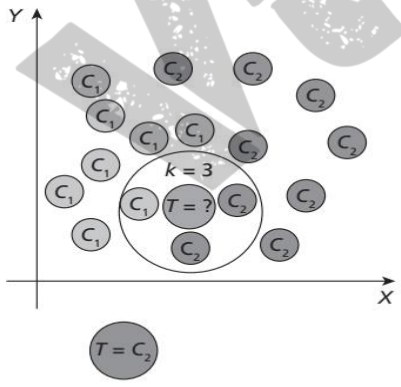


Figure 4.1: Visual Representation of *k*-Nearest Neighbor Learning

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.

Algorithm 4.1: k-NN

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.

4.3 Weighted k-Nearest-Neighbor Algorithm

The weighted KNN is an extension of k-NN.It chooses the neighbors by using the weighted distance. In weighted kNN, the nearest k points are given a weight using a function called as the kernel function. The intuition behind weighted kNN, is to give more weight to the points which are nearby and less weight to the points which are farther away.

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.

4.4 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.

4.5 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 T,

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,

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

Where, τ is called the bandwidth parameter and controls the rate at which w_i reduces to zero with distance from x_i.

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:

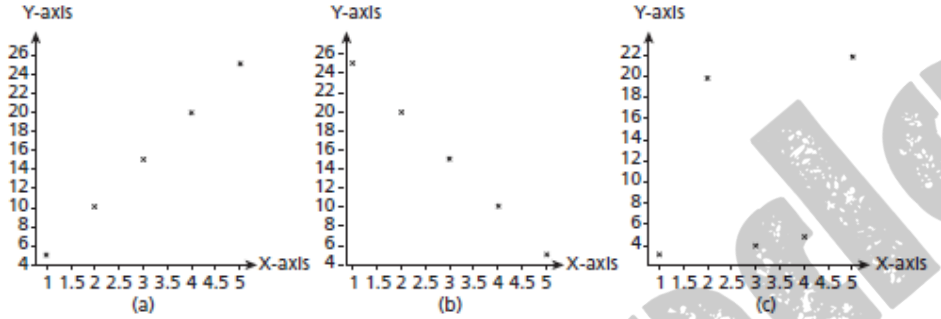


Figure 5.1: Examples of (a) Positive Correlation (b) Negative Correlation (c) Random Points with No Correlation

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.

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.

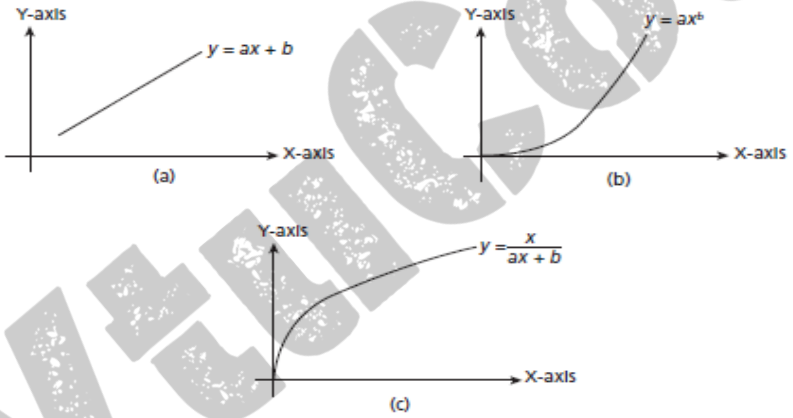


Figure 5.2: (a) Example of Linear Relationship of the Form $y = ax + b$ (b) Example of a Non-linear Relationship of the Form $y = ax^b$ (c) Examples of a Non-linear Relationship $y = \frac{x}{ax + b}$

Types of Regression

Types of Regression Methods

The classification of regression methods is shown in Figure 5.3.

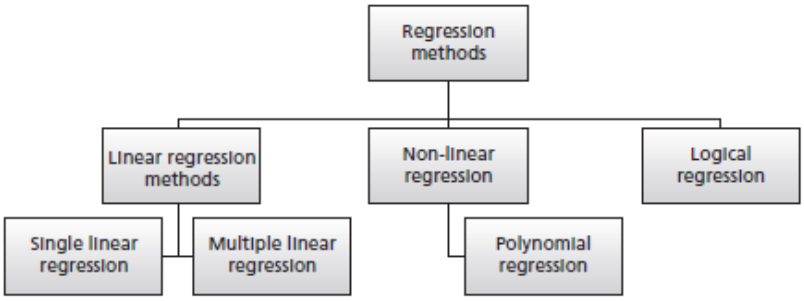


Figure 5.3: Types of Regression Methods

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.

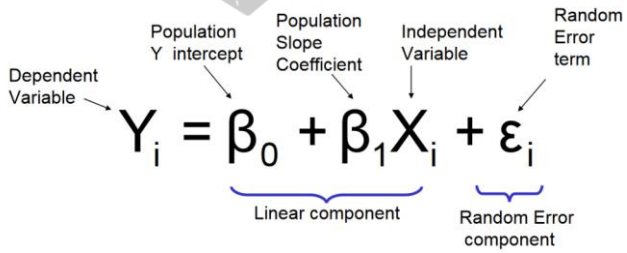
Objective Function: Ridge regression adds an L2 penalty to the linear regression loss function: Ridge = RSS + λΣ(βi^2), where RSS is the residual sum of squares, λ is the regularization strength, and (βi^2) represents the square of the coefficients.

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:



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 $\mathbf{y}_i$ 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.

$$\mathbf{y = b0 + b1x1 + b2x2 + \ldots + bnxn + e}$$

- 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

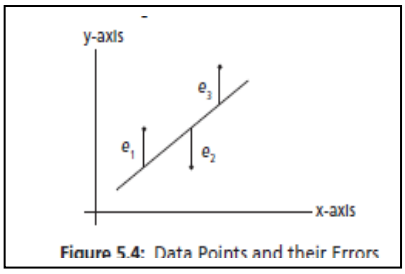


Figure 5.4: Data Points and their Errors

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

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

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.

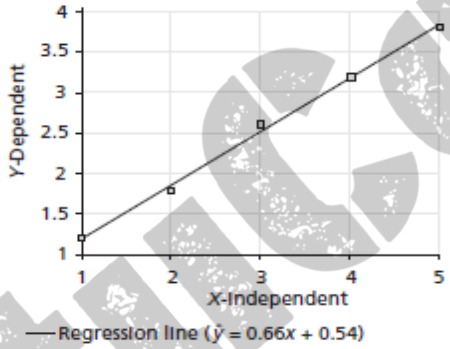


Figure 5.5: Linear Regression Model Constructed

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.

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:

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

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:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n-1}\left|y_i - \hat{y}_i\right| \tag{5.12}$$

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:

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

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.

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

CHAPTER 5

REGRESSION ANALYSIS

5. Consider the following dataset in Table 5.11 where the week and number of working hours per week spent by a research scholar in a library are tabulated. Based on the dataset, predict the number of hours that will be spent by the research scholar in the 7th and 9th week. Apply Linear regression model.

Table 5.11

x_i (week)	**1**	**2**	**3**	**4**	**5**
y_i (Hours Spent)	**12**	**18**	**22**	**28**	**35**

Solution

The computation table is shown below:

x_i	y_i	$x_i \times x_i$	$x_i \times y_i$
1	**12**	**1**	**12**
2	**18**	**4**	**36**
3	**22**	**9**	**66**
4	**28**	**16**	**112**
5	**35**	**25**	**175**
Sum = 15 **avg(** x_i **)=15/5=3**	**Sum = 115** **avg(** y_i)=115/5=23	**Avg (** $x_i \times x_i$)=55/5=11	**Avg(** $x_i \times y_i$)=401/5=80.2

The regression Equations are

$$a_1 = \frac{\left(\overline{xy}\right) - \left(\overline{x}\right)\left(\overline{y}\right)}{\left(\overline{x_i^2}\right) \times \left(\overline{x}\right)^2}$$

$$a_0 = \left(\overline{y}\right) - a_1 \times \overline{x}$$

$$a_1 = \frac{80.2 - 3(23)}{11 - 3^2} = \frac{80.2 - 69}{11 - 9} = \frac{11.2}{2} = 5.6$$

$$a_0 = 23 - 5.6 \times 3 = 23 - 16.8 = 6.2$$

Therefore, the regression equation is given as

$$y = 5.6 + 6.2 \times x$$

The prediction for the 7th week hours spent by the research scholar will be

$y = 5.6 + 6.2 \times 7 = 49$ hours

The prediction for the 9th week hours spent by the research scholar will be

$y = 5.6 + 6.2 \times 9 = 61.4 \approx 61$ hours

6. The height of boys and girls is given in the following Table5.12.

Table 5.12: Sample Data

Height of Boys	65	70	75	78
Height of Girls	63	67	70	73

Fit a suitable line of best fit for the above data.

Solution

The computation table is shown below:

x_i	y_i	$x_i \times x_i$	$x_i \times y_i$
65	**63**	**4225**	**4095**
70	**67**	**4900**	**4690**
75	**70**	**5625**	**5250**
78	**73**	**6084**	**5694**
Sum = 288 Mean(x_i)=288/4=72	**Sum = 273 Mean(y_i**)=273/4=68.25	**Avg ($x_i \times x_i$**)=20834/4=5208.5	**Avg($x_i \times y_i$**)=19729/4=4932.25

The regression Equations are

$$a_1 = \frac{\left(\overline{xy}\right) - \left(\bar{x}\right)\left(\bar{y}\right)}{\left(\overline{x_i^2}\right) \times \left(\bar{x}\right)^2}$$

$$a_0 = \left(\bar{y}\right) - a_1 \times \bar{x}$$

$$a_1 = \frac{4932.25 - 72(68.25)}{5208.5 - 72^2} = \frac{18.25}{24.5} = 0.7449$$

$$a_0 = 68.25 - 0.7449 \times 72 = 68.25 - 53.6328 = 14.6172$$

Therefore, the regression line of best fit is given as

$$y = 0.7449 + 14.6172 \times x$$

7. Using multiple regression, fit a line for the following dataset shown in Table 5.13. Here, Z is the equity, X is the net sales and Y is the asset. Z is the dependent variable and X and Y are independent variables. All the data is in million dollars.

Table 5.13: Sample Data

Z	**X**	**Y**
4	12	8
6	18	12
7	22	16
8	28	36
11	35	42

Solution

The matrix X and Y is given as follows:

$$X = \begin{pmatrix} 1 & 12 & 8 \\ 1 & 18 & 12 \\ 1 & 22 & 16 \\ 1 & 28 & 36 \\ 1 & 35 & 42 \end{pmatrix}$$

$$Y = \begin{pmatrix} 4 \\ 6 \\ 7 \\ 8 \\ 11 \end{pmatrix}$$

The regression coefficients can be found as follows

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

Substituting the values one get,

$$\hat{a} = \left(\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 12 & 18 & 22 & 28 & 35 \\ 8 & 12 & 16 & 36 & 42 \end{pmatrix} \times \begin{pmatrix} 1 & 12 & 8 \\ 1 & 18 & 12 \\ 1 & 22 & 16 \\ 1 & 28 & 36 \\ 1 & 35 & 42 \end{pmatrix} \right)^{-1} \times \begin{pmatrix} 1 & 12 & 8 \\ 1 & 18 & 12 \\ 1 & 22 & 16 \\ 1 & 28 & 36 \\ 1 & 35 & 42 \end{pmatrix}^T \times \begin{pmatrix} 4 \\ 6 \\ 7 \\ 8 \\ 11 \end{pmatrix}$$

$$= \begin{pmatrix} 5 & 115 & 114 \\ 115 & 2961 & 3142 \\ 114 & 3142 & 3524 \end{pmatrix}^{-1} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 12 & 18 & 22 & 28 & 35 \\ 8 & 12 & 16 & 36 & 42 \end{pmatrix} \times \begin{pmatrix} 4 \\ 6 \\ 7 \\ 8 \\ 11 \end{pmatrix}$$

$$= \begin{pmatrix} -0.4135 \\ 0.39625 \\ -0.0658 \end{pmatrix}$$

Therefore, the regression line is given as

$$y = 0.39625x_1 - -0.0658x_2 - 0.4135$$

***