

```
toyPuzzleTypeCostRDD = sc.parallelize([Row (toy_type =
"puzzle_Garden", puzzle_code= "10725", pieces = 100,
puzzle_cost_USD = 1.35) ]
```

[Created row toyPuzzleTypeCost RDD with a tuple using parallelize method of Spark Context, sc]

```
toyPuzzleTypeCostSchemaRDD=    HiveContext(sc).inferSchema
(toyPuzzleTypeCostRDD)
```

[Created row-schema, toyPuzzleTypeCostSchemaRDD using method, inferSchema of Hive Context inbuilt in Spark Context, sc]

```
toyPuzzleTypeCostSchemaRDD.registerTempTable
("toyPuzzleTypeCostTbl")
```

[Created RDD using registerTempTable method for the row object schema. Table toyPuzzleTypeCostTbl creates toyPuzzleTypeCostSchemaRDD.]

Numeric Operations on RDD Spark provides several descriptive statistics operations on RDDs

containing numeric data. Table 5.6 provides description of numeric RDD operations available in Spark.

Table 5.6 Numeric RDD Operations

Method	Description
count()	Returns the number of elements in an RDD. Returns BIGINT of 8 bytes. For example, Command toyPuzzleTypeCostTbl.count() returns 12 in the table in Example 5.10.
sum()	Returns the sum of the elements
mean()	Returns the sum of the elements divided by the number of elements
min()	Returns the minimum value of the elements
max()	Returns the maximum value of the elements
stdev()	Returns the statistical parameter, standard deviation of the elements
sampleStdev()	Returns the statistical parameter, standard deviation of a sample of the elements
variance()	Returns the statistical parameter, variance of the elements

sampleVariance()	Returns the statistical parameter, variance of a sample of the elements
------------------	---

Shared Variables The variables in HDFS have multiple copies in the data nodes of a cluster. While mapping or reducing functions for execution on a cluster node, the operations take place on separate copies of all variables used in the function. These copies also updated, but the copies or changes do not pass on to the driver program. Usages of read and write instructions for the variables are inefficient.

Broadcast variables and accumulators enable the implementation of shared variables.

Broadcast Variables Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost. Broadcast variables are created from a value denoted by a variable v and running method `sc.broadcast(v)`. The broadcast variable is a wrapper around v .

EXAMPLE 5.11

Recapitulate Example 5.8 (ii) of a toy company selling Jigsaws. The RDD has two lines after `filter()`. How will you use the intermediate results and execute action commands `count()` and `first()`?

SOLUTION

Recapitulate Example 5.10. Python code accesses the value of the broadcast variable as follows:

```
>>> broadcastVar = sc.broadcast([("puzzle_Garden", "10725", 100, 1.35)])
```

The result will be: <pyspark.broadcast.Broadcast object at 0x102789f10>. The value can be accessed by calling the `value` method as follows:

```
>>> broadcastVar.value
```

Result will be `[("puzzle_Garden", "10725", 100, 1.35)]`

Accumulators

Spark supports accumulators of numeric type. A programmer can provide additional support to other data types. Accumulators are special variables. They add the values using associative and commutative operations. They also support parallel run: for example, in `count()` or `sum()`.

An accumulator variable creates from an initial value v . Later the value accumulates into that using `+=` operator. For example, in `count()` initial value is 0. The command to create accumulating variable is `sc.accumulator(v)`.

EXAMPLE 5.12

Recapitulate Example 5.10. How does Python code access the value of the accumulator variable?

SOLUTION

Python code accumulates the value using variable `v` as follows:

```
>>> accumColumns = sc.broadcast(0)
>>> accumColumns
```

The result will be <Accumulator id = 0, value = 0>.

When following command executes:

```
>>> sc.parallelize([1.35, 2.85, 1.37]).foreach(x:
    accumColumns.add(x))
```

The result will be 5.57.

5.4.3 Machine Learning with MLlib

Apache MLlib is a component of Spark that is open source downloadable from Apache Spark website. Figure 5.8 shows the main usages of MLlib (machine learning library).

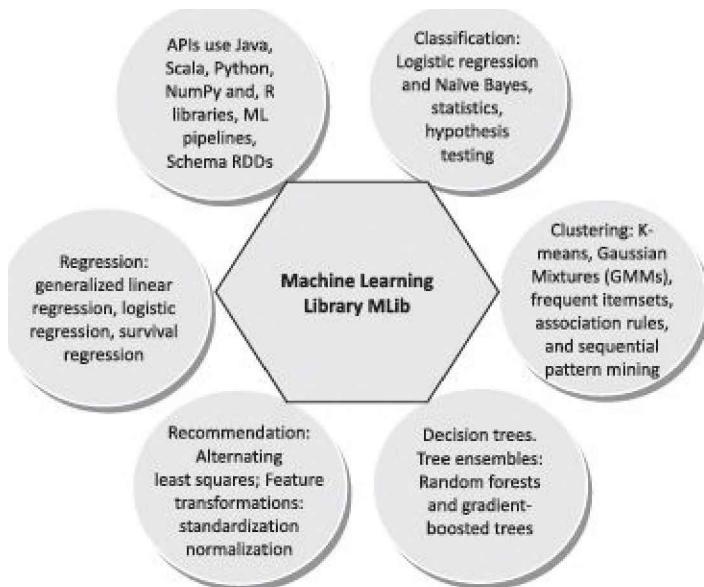


Figure 5.8 Main usages of MLlib machine learning library

Spark Support ML pipelines. An ML pipeline means data taken from data sources, passes

through the machine learning programs in between and the output becomes input to the application. Decision tree, knowledge discovery, clustering and mining are examples of ML pipeline application.

Spark 2.3 supports Python UDFs, VUDFs, block-level UDFs with block-level arguments and return types, complex object types (array map and structure), and conversions or transformations of object types. These features are widely used for the ML applications.

ML datasets are RDDs, thus use HDFS, HBase or local files. MLlib APIs are interoperable with Spark SQL. MLlib Python implementation adds Python APIs. MLlib interoperates with NumPy in Python.

Chapter 6 will describe ML algorithms in detail.

Self-Assessment Exercise linked to LO 5.3

1. What are the steps for downloading a Spark version?
2. List the Spark supported file formats and their usages.
3. Write the use of transform commands in RDD programming, filter(), map(), mapValues(), flatMap(), sort(), partitionBy(), groupByKey(), reduceByKey(), aggregateByKey(), pipe(), coalesce(), sample(), union(), join(), cogroup(), and crossProduct().
4. How do broadcast variables and accumulators enable the implementation of shared variables?
5. Create, using inferSchema method of HiveContext inbuilt in Spark Context sc, row-schema toPuzzleTypeCostSchemaRDD.
6. What does machine learning mean? What are the main usages of MLlib?

5.5 . DATA ETL (EXTRACT, TRANSFORM AND LOAD) PROCESS

The ETL process combines the following three functions into one:

LO 5.4

ETL processes using built-in
functions and operators,
and ETL pipelines

1. Extract which does the acquisition of data from Data Store querying or from another program,
2. Transform which does the change of data into a desired file, columnar, tabular or other. Transformation converts the previous form of the extracted data into a new form. Transformation occurs by using rules or lookup

tables. Transformation uses the functions, namely join(), groupBy(), cogroup(), filter(), map(), mapValues(), flatMap(), sort(), partitionBy(), groupByKey(), reduceByKey(), aggregateByKey(), pipe(), coalesce(), sample(), union(), crossProduct(). Spark 2.3 includes transformation functions on complex objects like arrays, maps and set of columns. Pandas provide powerful transformation UDFs, VUDFs and GVUDFs.

- Load which does the process of placing transformed data into another Data Store or data warehouse for usage by an application or for analysis.

Python, Spark SQL and HiveQL support ETL programming and extracting by query-processing and text processing.

5.5.1 Composing Spark Program Steps for ETL

Spark 2.3 ETL Pipeline Apache Spark 2.3+ includes usage of UDFs, VUDFs and Data Source API v2. These facilitate the creation of ETL pipelines easily. Figure 5.9 shows an ETL pipeline using Spark SQL for ETL Process and Data Source API v2 in Spark 2.3.

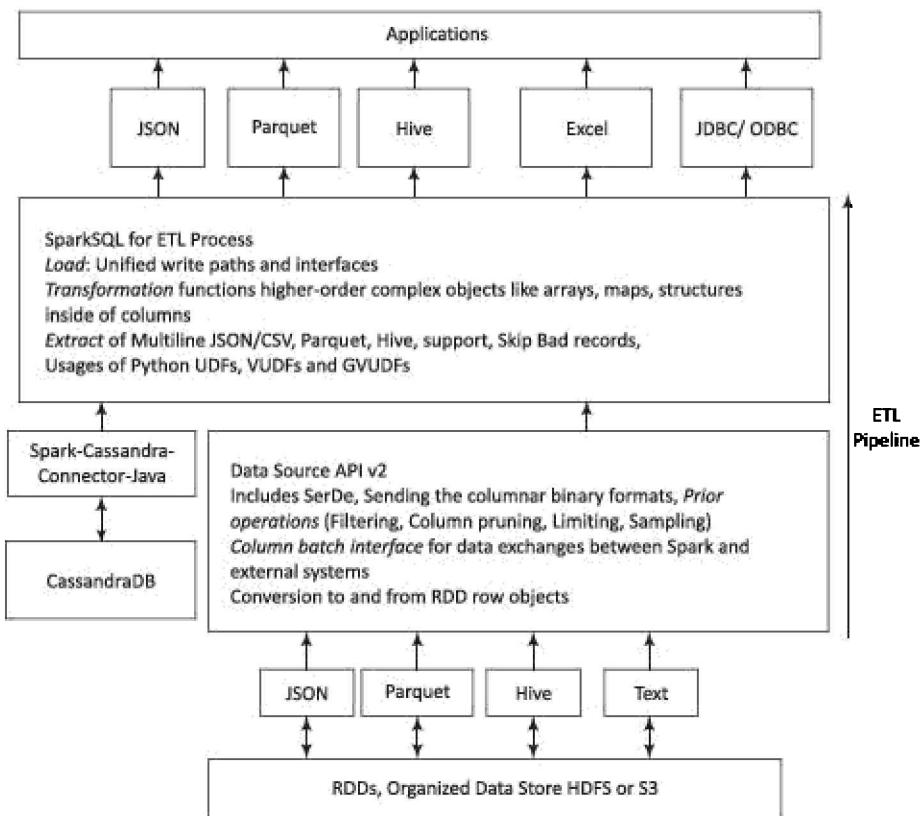


Figure 5.9 ETL pipeline using Spark SQL

Extract

Skipping Corrupt or Bad Records or Files A record can sometimes be bad. For example, “toyPuzzle_Type”, toyPuzzle_Airplane. Here the data type in schema is string, but quotes are missing in toyPuzzle_Airplane. Use the following command to ignore bad records or file: `spark.sql.files.ignoreCorruptFiles = true`.

Spark supports parser for text file formats in JSON and CSV with three modes: PERMISSIVE, DROPMALFORMED OR FAILFAST. Reading data requires a parser in a SELECT for querying or load for extraction process.

Extract and Load: Multi-line JSON/CSV Support *Load and Save files:* SerDe uses codes for obtaining records from unstructured data. Save process uses serializer codes and Loading (extracting) process uses deserializer.

The following example explains the codes for sequence File, JSON and CSV file load and save functions for obtaining records/rows/files.

EXAMPLE 5.13

Using Python,

- (i) How does a sequence file load?
- (ii) How does a text file ‘jigsaw_puzzle_info.txt’ load? (Example 4.10)
- (iii) How does a file productCodesCosts save as a TextFile in Python?
- (iv) How does a JSON studentSemGrades file load?
- (v) Assume record field names are “StuName”, “rollNum”, “enrollNum”, “SemID”, “courseID”, “subjectID”, “subjectType”, “SemSubjGrade” in a CSV file line. How does StuSemRecord(line) load from a JSON file ‘studentSemGrades’?
- (vi) How does StuSemRecord(line) save as CSV format file?

SOLUTION

- (i) Load a sequence file using the following statement:

```
val           data=           sc.sequenceFile(inFile
"org.apache.hadoop.io.Text",
"org.apache.hadoop.io.IntWritable")
```

- (ii) Load a text file jigsaw_puzzle_info.txt using the following statement:

```
hiveCtx.textFile
(file://home/PySpark/hive/‘jigsaw_puzzle_info.txt’)
```

(iii) Save the text file productCodesCosts using the following statement:

```
productCodesCosts.saveAsTextFile(outputFile)
```

(iv) Load an unstructured JSON file studentSemGrades using the following statement:

```
import json

studentGradesData = input.map(hiveCtx:
    json.loads(studentSemGrades))
```

(v) Load StuSemRecord(line) from a CSV file 'studentSemGrades' using the following statement:

```
import stringIO
import csv

..
.

def StuSemRecord(line): """Parse a CSV line"""

reader = csv.DictReader(input, fieldNames = ["StuName",
"rollNum", "enrollNum", "SemID", "courseID", "subjectID",
"subjectType", "SemSubjGrade"])

return reader.next()

input = hiveCtx.textFile(inputFile).map(StuSemRecord)
```

(vi) Save StuSemRecord(line) from a CSV file 'studentSemGrades' using following statement:

```
def writeStuSemRecords(records): """Write a CSV line"""

output =scv.DictWriter(output, fieldNames = ["StuName",
"rollNum", "enrollNum", "SemID", "courseID", "subjectID",
"subjectType", "SemSubjGrade"])

for record in records
writer.writerow(record)

return [output.getvalue()]

hiveCtx.mapPartitions(writeStuSemRecords).saveAsTextFile
(outputFile)
```

Transformation The following example explains Spark SQL transformations in Spark 2.3. The example shows complex objects, nested tables (one column rows) and array transformations and uses the DataframeWriter API.

EXAMPLE 5.14

Use Spark 2.13 SQL APIs and DataframeWriter API.

- (i) How does a table-column schema create for a nested table (single Column table) `tbl_puzzleCost`? (Example 5.1) Text file is ‘`jigsaw_puzzle_info.txt`’. (Example 4.10)
- (ii) How does the value filter using a row from table ‘`tbl_puzzleCost`’ for a puzzle costing above USD 1.00? (Example 5.1)
- (iii) How does `yearlySales` compute from table for Jaguar Land Rover Yearly sales, JLRDS (Example 5.2) from `CarShowroomsCummulativeYearlySales` table?
- (iv) CREATE Hive-SerDe table. Create a table similar to one in Example 5.1. Use DataframeWriter API.
- (v) CREATE Hive-SerDe table. Create a table similar to one in Example 5.1. Use Hive.

SOLUTION

- (i) Table schema for a nested table (single Column table) `tbl_puzzleCost` creates:

```
FROM tbl_puzzleCost;  
tbl_nested  
| -- key: string (nullable = false)  
| -- values: array (nullable = false)  
| -- element: float (containsNull = false)
```

- (ii) The following Spark SQL statement filters an array or row or nested table, `tbl_puzzleCost`:

```
SELECT FILTER(values, v4 -> v4 > 1.00) AS v4  
FROM tbl_puzzleCost;
```

- (iii) The following Spark SQL aggregation function for the sum of an array or row or nested table elements, JLRDS in `CarShowroomsCummulativeYearlySales` row `tbl_JLDR`:

```
SELECT REDUCE (values, (value, JLRDS) -> value + JLRDS)
AS yearlyJLRSales FROM tbl_JLRDS;
```

- (iv) The following statements create a Hive-SerDe table using the DataframeWriter:

```
CREATE Hive-serde tables
df.write.format("hive")
.option("fileFormat", "avro")
.saveAsTable("tbl_puzzleTypeCodeCosts")
CREATE TABLE tbl_puzzleTypeCodeCosts (puzzleType STRING,
puzzleCode STRING, puzzlePieces SMALLINT, puzzleCost
FLOAT)
STORED AS ORC
```

- (v) The following statements create a table using Hive:

```
CREATE TABLE tbl_puzzleTypeCodeCosts (puzzleType STRING,
puzzleCode STRING, puzzlePieces SMALLINT, puzzleCost
FLOAT)
USING hive
OPTIONS (fileFormat 'ORC')
```

Self-Assessment Exercise linked to LO 5.4

1. What are the three ETL functions in Data Store?
2. How do line records *extract* from a multiline JSON file?
3. How do values filter from a column ‘puzzleCost’ for puzzle costing below USD 1.00?
4. What are the program steps for creating a Hive-SerDe table using the DataframeWriter?

5.6 • INTRODUCTION TO ANALYTICS, REPORTING AND VISUALIZING

“Analytics is the discovery, interpretation, and communication of meaningful patterns in data.” Since Big Data analytics require extensive computations, the

algorithms and software used for analytics combine the most current methods of computer science, statistics and mathematics.” (Wikipedia)

LO 5.5

Analytics, data/information reporting and visualizing methods

Analytics needs interpretation, reporting of meaningful patterns and gaining insight into new knowledge. Visualization is an effective method for examining the interpretations of reports. The subsections ahead introduce analytics, reporting and visualization methods.

5.6.1 Introduction to Analytics

Some examples of open source tools for analytics are Python, R, Apache Spark, Apache Storm, Pig and Hive. Examples of commercial tools are SAS, Tableau, Excel, QlikView and Splunk. Figure 5.10 shows a framework for applications and analytics using Spark.

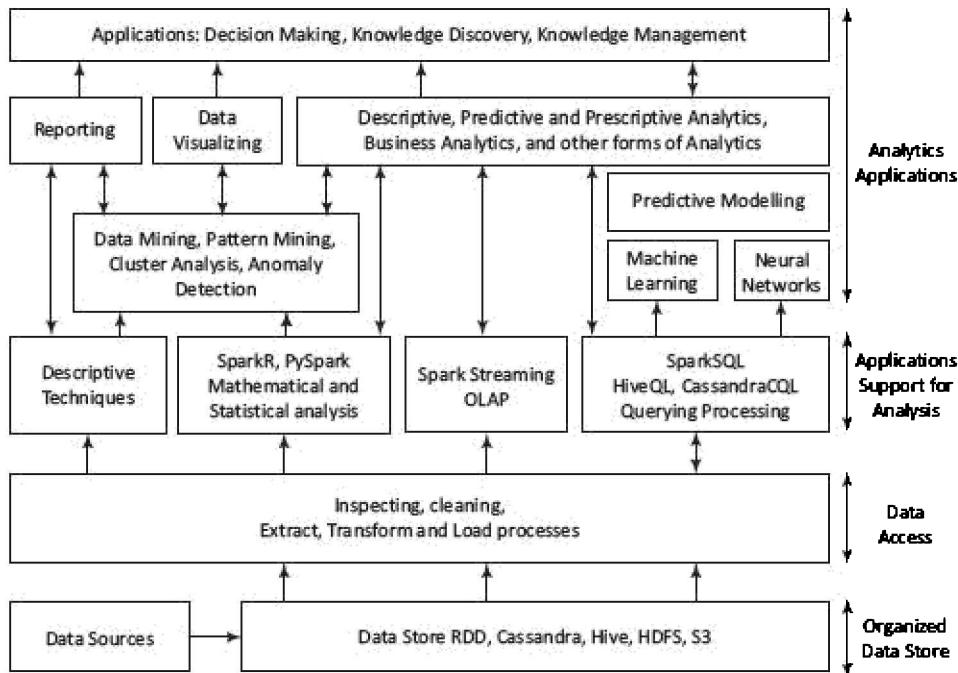


Figure 5.10 Processing framework for applications and Big Data analytics using Spark

Spark Stack provides support to applications for analysis of huge data from multiple sources and Data Stores. Analytics also use machine learning and neural networks, which enable predictive modeling and decisions. Analytics use data mining, pattern mining, cluster analysis and detect anomalies.

Applications use the results derived from the descriptive, predictive and prescriptive analytics, business analytics and other forms of analytics. The applications also use the

results derived from reporting and visualizations. Common applications are decision making about further actions required, knowledge discovery and knowledge management.

The example below explains the usages of analytics in different domains.

—

EXAMPLE 5.15

- (i) How does analytics help a manufacturing company for toys and puzzles?
- (ii) How does analytics helps a company for chocolates, which sells its products through ACVMs?
- (iii) How does analytics help a manufacturing, sales and service company for cars of different models?

SOLUTION

- (i) An analysis of sales figures for toys and puzzles, lets the toy company understand the preferences of children in different age groups, regions and income groups. The knowledge of preferences enables appropriate manufacturing, supply, sales, quality enhancement and promotion activities. The predictive analytics enable the company to take steps from the forecast about future growth, estimate the direction for future innovations in children toys and puzzles, and plan for the desired products mix in future.

The company organizes quizzes in schools of science, mathematics, geography and other subjects and awards brilliant students, as a part of promotional activity. The company also analyses the impact on sales after this activity.

- (ii) Analysis of sales figures and machine-users data by the company for chocolates being sold through ACVMs lets the company understand the following: (a) users and their needs, (b) chocolate preferences for specific flavours among children and youth in different regions in the country, different areas of cities, age groups, and sales in specific periods in a year. Detailed analysis enables the appropriate manufacturing, supply-chain, sales, innovations and promotional activities.

The predictive analytics enable the company to know the predictions about growth in profits, sales, identify future installation-areas for ACVMs, design relocation strategy and understand directions of future innovations and promotional strategy, needs, and desired product-mix of the company. The company also analyzes the impact of rewarding chocolate buyers on birthdays, marriage anniversaries and festival periods. Company plans incentives for frequent buyers. The company also organizes quizzes in schools and colleges.

Company rewards brilliant school students as a part of promotional strategy.

- (ii) Analysis of sale figures, customer, and service center feedbacks by the company for cars in different models, lets the company understand the followings: (a) users preferences for the different models and colour shades, (b) car model's sales in different regions in the country, age groups, and sales in specific periods in a year. Detailed analysis enables the appropriate product mix, manufacturing, supply-chain management, competition, advertisement strategy, innovations in design and shades, and facilitates promotional activities.

Predictive analytics enables company to know the predictions about growth in profits, sales and competition; identify car showrooms future expansion in different regions, and understand the direction for future innovations, such as use of IoT for studies on functioning of car components and IoT based maintenance services. The company can plan future expansion in plant machinery and ideal future product-mix strategy of the company. The company can also analyze the impact of incentives to showrooms for higher sales growth, festive-period incentives to buyers, and organizing the cultural events in order to design appropriate promotional activities.

5.6.2 Data/Information Reporting

Reports are essential after any analysis. Some important reasons for generating reports are:

1. Provide cross-databases and the data sheets [Cross-database created by application/system software so as to enable access to different database formats.]
2. Enable accesses to multi-business system data
3. Provide multi-data source correlations
4. Present related businesses data integrated in a Data Store
5. Enable more data applications for business control and operations analysis.

Data Organization Guidelines for Reports Some guidelines for organizing data in reports are:

1. Must have abstract, context introduction and conclusion sections
2. Divide the contents in sections or split it into multiple sheets with one context in one sheet
3. Place similar items under the same heading or in the same column in case of

spreadsheets

4. Insert figures, charts or graphical plots to enhance the readability of the report
5. Position critical data appropriately
6. Avoid blank text areas or rows and columns in case of sheets.

Data Format Guidelines Some guidelines for formatting data in reports are:

1. Use section and subsection headings or column labels to identify contents
2. Use appropriate font, alignment, format, pattern, border or capitalization style
3. Use text borders or cell borders to distinguish data
4. Maintain same formats for paragraphs, section headings, subsection headings and captions of table and figures.

Report Designer A report designer has three components: (i) report, (ii) Data Tools Platform/Web Tools Platform (DTP/WTP), and (iii) chart designers. Users can add a custom designer at their end. The designer outputs to process at a design engine, which communicates the outputs in XML format to the Report Engine. Figure 5.11 shows the features of the components of a report designer.

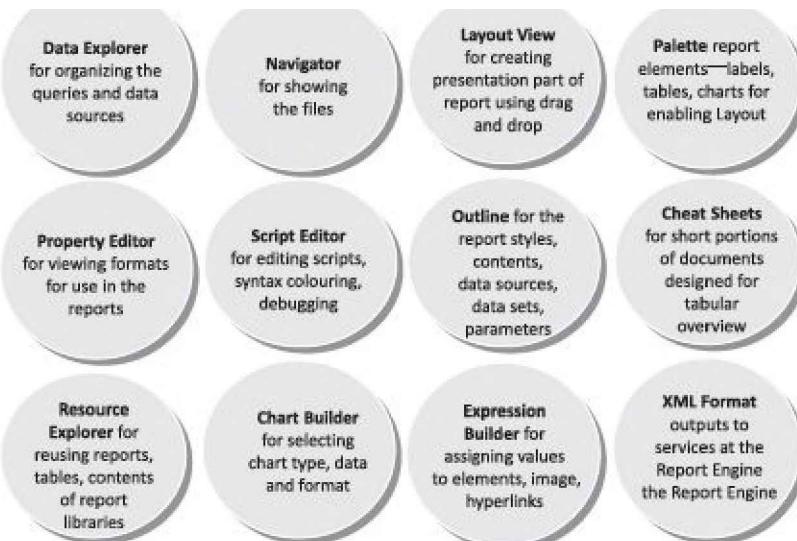


Figure 5.11 Features of the components of report designer

Open Source Eclipse Business Intelligence and Reporting Tool (BIRT)

BIRT is an open source software project created by the Eclipse Foundation. Its latest version is 4.8.0 was released in June 2018. BIRT includes Eclipse Report Designer (ERD),

Eclipse Report Engine (ERE) and Eclipse Charting Engine (ECE). BIRT creates reports and enables data visualization of the reports. Reports can be embedded into rich Java, Java EE client and web applications.

Figure 5.12 shows the architecture of BIRT.

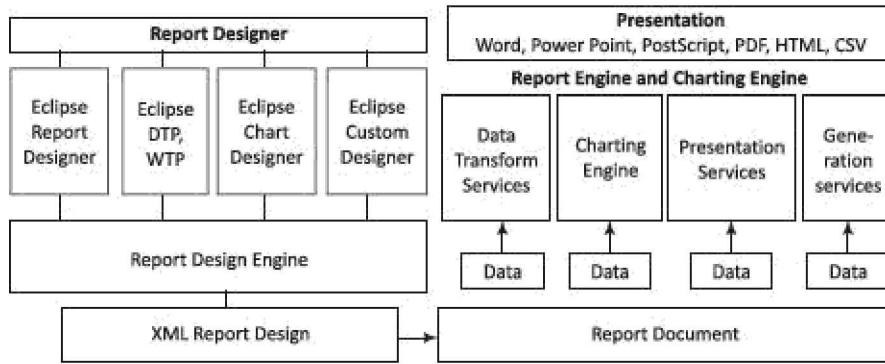


Figure 5.12 BIRT architecture design and report engine components

Data explorer creates views of multi-dimensional cubes from the datasets provided. Cubes enable building of dynamic cross-tables. A dimension in the view can be used to design report parameters.

Report Engine The figure shows that report document and data process occurs at four service plug-ins – data transformation, charting, presentation and generation services.

Transformation service plug-in presents the data after sorting, summarizing, filtering and grouping. The transformations are performed as per the needs of user. A program function performs the transformation in the ETL process, but BIRT does that for simple data sources, such as Java objects or flat files. BIRT does provision for complex operations, such as grouping on sums, percentages of overall totals.

Charting services at the engine generate the charts. The Charting Engine is used to design and generate the charts. A chart can be separate or embedded within the generated reports, such as BIRT reports. Charting engine API (CE API) in Java/Java EE adds charting capabilities to their applications. The design and report engines make use of CE API for generating the charts.

BIRT ‘viewer’ enables preview reports at an Apache Tomcat server included in the BIRT. The server fulfills the client request for preview. The outputs are web output as a single HTML document, paginated HTML, PDF, XLS, DOC, PPT, and Postscript. Additionally, the viewer has functionality for exporting the data to CSV, printing and Table of Contents.

5.6.3 Data Visualization

Data visualization is a technique that entails the creation and study of the visual representation of data, meaning “information that has been abstracted in some schematic form, including attributes or variables for the units of information”. A primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics. Data visualization tools have shifted the interpretation of data from dashboard displays to quick digestion of real-time analysis and custom analysis. (*Wikipedia*)

Data Display can be accessed via PC dashboards or mobile terminals and interpreted thereafter. Display improves the reading of reports. Reports include the functions for display of the analysis of a wide variety of charts, multi-dimensional analysis, real-time analytics, and slicing and dicing views of reports. Data visualization tools have completely changed the concept dashboard displays for interpretations.

Data visualization uses techniques which data or information encodes in visual objects such as points, lines or bars. Communication of information using visual objects to viewers is more efficient and clear. Data visualization relates to a number of fields: information and statistical graphics, scientific information visualization and exploratory data analytics. Visualization communicates clearly and stimulates the viewer's attention and engagement.

Table 5.7 describes data visualization tools and their usages.

Table 5.7 Data visualization tools and their usages

Data Visualization Tool	Description
matplotlib	Python matplotlib for plotting mathematical functions. Several libraries such as vispy, bokeh, seaborn, pygal, folium build up on matplotlib for data visualization.
Chart.js	A tiny widely used open source library that supports just six chart types: line, bar, radar, polar, pie and doughnut. Employs HTML5 canvas element for rendering charts. Not used when the application is big and complex.
Google Charts	Shows charts in HTML5/SVG and many charts including interactive charts and most commonly used chart types like bar, area, pie and gauges.
Tableau	Public version is open source. Tableau is an easy-to-learn popular data visualizing tool. Tableau communicates insights through data visualization. The charts easily embed in any web page. Supports a wide variety of charts, graphs, maps and other graphics. Tableau's visuals enable quick investigation of a hypothesis. Commercial version includes additional facilities for handling over a million rows.
D3.js	D3 (Data Driven Documents) is free and open source, deploys HTML, CSS and SVG, and shows the amazing charts and diagrams. It has rich features and is quite interactive.

FusionCharts	Includes over 90+ chart types, 965 maps, collection of business dashboards and live demos. The charts and maps are highly customizable and have attractive interactions. The tool supports to (i) JSON and XML data formats and (ii) export the charts in PNG, JPEG, SVG or PDF formats. FusionCharts is a commercial tool.
QlikView	Is a fast processing tool with usages for the intuitive GUIs, interactive plotting of data, slicing and dicing features in data visualizing, and creating highly useful visualizations and dashboards.
Splunk	Is a machine log analytic tool. It includes powerful visualization features and easy to use tool due to its web interface. Features are events list, table visualization, charts, gauges, maps, Trellis layout for visualizations, dashboard creation with XML and Drilldown and dashboard interactivity.

Self-Assessment Exercise linked to LO 5.5

1. What do the terms analytics and analysis mean?
2. List the functions of Report Designer.
3. List the services of the Report Engine.
4. Explain events list, table visualization, charts, gauges, maps, and Trellis layout for visualization.

KEY CONCEPTS

accumulator

ad hoc query

aggregation

analytics

BIRT

broadcast

Ceph

data visualization

DataFrame

data pipeline

data source

dicing

drill down analysis

ETL

ETL pipeline

GraphX

GroupBy

grouped VUDF

HDFS

HiveContext

Hive Server

immutability

JDBC

matplotlib

Mesos

metadata

metastore

MLib

ML pipeline

nested table

NumPy

ODBC

Outliers

Pandas

Parquet

PySpark

Python

query processing

RDD

RDD programming

Report Designer

Report Engine

S3
SchemaRDD
SciPy
SequenceFile
SerDe
slicing
Spark
SparkContext
Spark SQL
Spark Streaming
Tableau
transform
UDF
Vectorized UDF
Viewer



LO 5.1

1. Apache® Spark™ is a *fast and general compute engine*. Apache® Spark™ powers analytics applications up to 100 times faster.
2. It supports HDFS compatible data. Spark has a simple and expressive programming model.
3. Spark standard API enables creation of the application APIs using Scala, Java, Python and R. Spark consists of software stack with Spark SQL, Spark Streaming, Spark Arrow, SparkR, MLlib and GraphX components.

LO 5.2

1. Spark SQL is a Spark module for structured data; SQL runs the queries on Spark data in traditional business analytics and visualization applications. Spark SQL enables Spark datasets to use JDBC API or ODBC API.

2. DataFrames can be created from different data sources. Examples of data sources are JSON datasets, Hive tables, Parquet row groups, structured data files, external Data Stores and RDDs.
3. Spark and Python provide powerful Big Data analysis tools. Python provisions Python NumPy, SciPy, Pandas consists of advanced functions for analytics, and provides an Integrated Development Environment (IDE).
4. Python Pandas library functions provide database style DataFrames, Merge, join, and concatenation of objects, RPy interface for R functions plus additional functions, SQL like features: SELECT, WHERE, GROUPBY, JOIN, UNION, UPDATE and DELETE, size mutability which means columns can be inserted and deleted from DataFrame and higher dimensional objects. Pandas GroupBy feature of split-apply-combine enables group vectorized UDFs.

LO 5.3

1. Spark 2.3.1. Spark runs on Windows and UNIX-like systems, Linux, Mac OS. Spark runs on Java 8+, Python 2.7+/3.4+ and R 3.1+. Spark 2.3.1 uses Scala 2.11 API when using compatible Scala version 2.11.x.
2. Spark Resilient Distributed Dataset (RDD) is a collection of objects distributed on many computing nodes. Each RDD can split into multiple partitions, which may be computed in parallel on different nodes of the cluster.
3. Spark RDD is immutable (not capable of or not susceptible to change). A new RDD creates on *transform* and *action* commands.
4. Spark supports Python UDFs, VUDFs, block-level UDFs with block-level arguments and return types, complex object types, array map and structure conversions or transformations.
5. Spark MLlib supports ML pipelines. MLlib is widely used for the ML applications.

LO 5.4

1. Data ETL has three Data Store functions – Extract, Transform and Load. Applications using Spark 2.3 with Spark Arrow use the ETL data pipeline between data sources and applications.
2. Extract does the acquiring of data from a Data Store (by querying or another program). Transform does the change of data into a desired format. *Transform* uses *join()*, *groupByKey()*, *cogroup()*, *filter()*, *map()*, *mapValues()*, *flatMap()*, *sort()*, *pratitionBy()*, *groupByKey()*, *reduceByKey()*, *aggregateByKey()*, *pipe()*, *coalesce()*,

`sample(), union(), crossProduct()`.

3. Load saves the transformed data into another Data Store for usage by an application or for analysis.

LO 5.5

1. Analytics is the discovery, interpretation and communication of meaningful patterns in data. Spark Stack provides support to applications for analysis of data from multiple sources and Data Stores. Analytics use machine learning and neural networks, which enable predictive modeling and decisions based on the data. Analytics use data mining, pattern mining, clusters analysis and detect anomalies.
2. Reporting of results of analysis using appropriate tools is an essential step. Eclipse Foundation created BIRT. BIRT includes Eclipse Report Designer (ERD), Eclipse Report Engine (ERE) and Eclipse Charting Engine (ECE).
3. Data visualization is a technique that entails the creation and study of data using visual representations such as charts, plots and graphics. Examples of data visualization tools are matplotlib, Chart.js, Google charts, Tableau, D3.js, FusionCharts, QlikView and Splunk.

Objective Type Questions

Select one correct-answer option for each questions below:

- 5.1 Spark uses for data storage (i) HDFS file system, (ii) Hadoop compatible data source, such as HDFS, (iii) HBase, Cassandra and Ceph, and (iv) Amazon S3. Spark standard API enables creation of the application APIs in (v) Scala, (vi) Java, (vii) Python, (viii) Pandas. Spark resources management can be at (ix) stand-alone server, and (x) at a distributed computing framework, such as YARN and Mesos.

- (a) all except iv, v and ix
- (b) all
- (c) all except viii, ix and x
- (d) all except iii, viii and ix

- 5.2 Spark Stack includes (i) Spark SQL, (ii) Spark Streaming, (iii) Spark Arrow, (iv) SparkR,
(v) MLlib, (vi) GraphX, (vii) SparkContext, (viii) HiveContext and Pig, (ix) Cassandra, and
(x) PySpark.

- (a) all except viii and ix
- (b) all except iii and v
- (c) all except viii and ix
- (d) ii, iii, vii, viii

5.3 Steps between acquisition of data from different sources, and applications of analyzed data, and applications support by Spark for the analyses are as follows: (i) Storage of data from the multiple sources after acquisition, (ii) Partitioning of tables, (iii) Data pre-processing, (iv) filtering unreliable, irrelevant and redundant information, (v) Extract, transform and load (ETL) for analysis using Python or Scala, (vi) Mathematical and statistical analysis of the data obtained after querying relevant data needing in the analysis, and (vii) applications of analyzed data; for example, descriptive, predictive and prescriptive analytics.

- (a) all except ii and iii
- (b) all except v and vi
- (c) all except i and iv
- (d) all except ii

5.4 A Parquet file consists of (i) row groups and (ii) column groups. (iii) Each row can have number of columns, but each column has only one column chunk. (iv) A page is a conceptualized unit in a column chunk. (v) Only one page can store in a chunk. (vi) ORC array has two columns, one for array size and other for array dimensions. (vii) Parquet format file consists of an extra column per nesting level.

- (a) all
- (b) i to v
- (c) i, iii, iv
- (d) all except vi and vii

5.5 When using HiveQL and Spark SQL, the aggregation functions can be used for (i) analysis and (ii) filtering. Hive aggregation functions consist of (iii) count(*) and count(expr); (iv) sum(col), (v) sum (DISTINCT col), (vi) Avg (col), (vii) avg (DISTINCT col), (viii) GroupBy, with (vii) DataFrames as input.

- (a) i, ii, iii, iv and vi
- (b) i to v
- (c) all except ii and viii
- (d) all except v and vii

5.6 NumPy provides (i) one-dimensional efficient containers of generic data, (ii) definitions of non-arbitrary data-types, (iii) easy integration with a wide variety of Data Stores, (iv) import, export (load/save) files, (iv) creation of arrays, (v) inspection of properties, copying, sorting, and reshaping, (vi) addition and removal of elements in the arrays, indexing, sub-setting and slicing of the arrays, (vii) scalar and vector mathematics (such as +, -, ×, ÷, power, sqrt, sin, log), (viii) recoil (round up to nearest int), and (ix) base (round down up to the nearest int), round (round to nearest integer).

- (a) iii to vii
- (b) ii to v
- (c) ii to viii
- (d) All except ix

5.7 Spark supports following file formats: (i) text files (ii) CSV, (iii) TSV, (iv) JSON, (v) doc file, (vi) XML, (vi) sequenceFiles (vii) ObjectFiles, and (viii) protocol buffers.

- (a) all except v
- (b) all except iii and viii
- (c) ii to iv and vi to vii
- (d) i, ii, iv, vi

5.8 An RDD (i) is a fault-tolerant abstraction, (ii) enables in-memory cluster computations, (iii) is mutable, (iv) partitioned distributed collection of objects, (v) enables efficient execution of non-interactive data mining, and (vi) RDDs create only through operations which are deterministic.

- (a) all except ii to iv
- (b) all except i, iii
- (c) all except vi
- (d) all except iii and v

5.9 Transform command examples are (i) filter(), (ii) reduce(), (iii) collect(), (iv) map(), (v) mapValues(), (vi) flatMap(), (vii) sort(), (viii) partitionBy(), (ix) groupByKey(), (x) reduceByKey(), (xi) sample(), (xii) union(), (xiii) join(), (xiv) cogroup(), (xv) crossProduct(). Action command examples are (xvi) aggregateByKey(), (xvii) pipe(), (xviii) coalesce(), (xix) count(), and (xx) first().

- (a) all except ii, iii, xi, xvii i
- (b) all ii, iii, xvi, xvii and xviii
- (c) all except iii and xiv
- (d) all xvi to xviii

5.10 DataFrames represent (i) ML datasets, thus uses HDFS, (ii) ML datasets, thus uses HBase or local files. (iii) MLlib APIs are interoperable with Spark SQL. (iv) MLlib Python implementation adds Python APIs. (v) MLlib does not interoperate with NumPy in Python.

- (a) i to iii
- (b) all except v
- (c) all except ii
- (d) all

5.11 SerDe uses codes for obtaining the records from (i) structured data. (ii) A data saving process uses deserializer codes and (iii) Loading (extracting) process uses serializer. Apache Spark 2.3+ provisions for (iv) UDFs, (v) VUDFs and (vi) Data Source API v1. (vii) ETL pipeline refers to data collected from (viii) in-between processing using a chain of function calls.

- (a) all except v and vi
- (b) all except i
- (c) all except ii, iii and iv
- (d) iv, v, vii and viii

5.12 Spark Stack provides the (i) support to applications for the analysis of data, (ii) unstructured multiple sources, (iii) Data Stores. The (iv) analytics use machine learning algorithms, (v) neural networks, (vi) descriptive modeling, and (vii) enables decisions. Analytics use the algorithms for (viii) data mining, (ix) pattern mining (x) clustering, and (xi)

anomaly detection.

- (a) all except ii, v and vi
- (b) all except v and ix
- (c) all except ii and vi
- (d) all except vi and xi

5.13 Process for preparing a report document and data consists of service plug-ins for (i) data transformation, (ii) charting, (iii) summarizing (iii) presentation, and (iv) generation services. Transform services plug-in presents the data after (v) sorting, (vi) filtering, and (vii) grouping.

- (a) all except ii, v and vi
- (b) all except iii
- (c) all except vii
- (d) all except vi and vii

5.14 Data visualizing tool Tableau is (i) open source completely, (ii) easy-to-learn data visualizing tool, (iii) communicates predictions through data visualization. (iv) The charts can be easily embed in any web page. (v) Supports a wide variety of charts and graphs, but not maps.

(vi) Tableau's visuals enables quick investigation of a hypothesis.

- (a) ii, iv and vi
- (b) all except iii
- (c) all except i
- (d) all except iii and vi

Review Questions

5.1 What are the features present in the Spark architecture that enable fast computations and usages of expressive programming model? **(LO 5.1)**

5.2 Describe the functions of Spark SQL, Spark Streaming and GraphX? **(LO 5.1)**

5.3 How do Spark and Python provide a powerful Big Data analysis tool? **(LO 5.2)**

5.4 How does Parquet file usage differ from the usages of RCFile and ORCFile? **(LO 5.2)**

5.5 How does DataFrame create from JSON datasets and Hive tables? **(LO 5.2)**

- 5.6 What are the aggregation commands provisioned in Spark SQL? **(LO 5.2)**
- 5.7 How do NumPy, SciPy and Pandas Python libraries provision for advanced functions for analytics, and create an integrated development environment (IDE)? **(LO 5.2)**
- 5.8 How does the Spark version download, install, and start the use of SparkContext? **(LO 5.3)**
- 5.9 How does the Spark Resilient Distributed Dataset (RDD) programming collect the objects?
(LO 5.3)
- 5.10 Explain method of creation of RDDs using the *transform* and *action* commands. **(LO 5.3)**
- 5.11 Describe the machine learning algorithms available in Spark MLib. **(LO 5.3)**
- 5.12 How does the Spark 2.3 with Spark Arrow enable the creation of the ETL data pipeline between data sources and applications? **(LO 5.4)**
- 5.13 How does a tool help in reporting of results of analysis? **(LO 5.5)**
- 5.14 What are the actions performed by the Eclipse Report Designer (ERD), Eclipse Report Engine (ERE) and Eclipse Charting Engine (ECE). **(LO 5.5)**
- 5.15 Define data visualization, statistical graphics, plots and information graphics. **(LO 5.5)**
- 5.16 How do the following visualization tools matplotlib, Chart.js, Google charts, Tableau, D3.js, FusionCharts, QlikView and Splunk differ? **(LO 5.5)**

Practice Exercises

- 5.1 List Spark stack components and give examples of their applications. **(LO 5.1)**
- 5.2 A company manufactures and sells cars through a large number of showrooms. Each car showroom keeps their records in a main table and transaction tables. Recapitulate table in Practice Exercise 3.3. Describe the steps for analyzing the sales from HDFS compatible files. **(LO 5.2)**
- 5.3 How does the Parquet file structure look like for a large number of student semester grade sheets? A semester grade sheet consists of University Name, Department Name, Student name, enrollment number, class, roll number, program of study, batch (for example 2017-2010), semester, subjects type (theory/practical/project/on-line), subject names, credit, credits awarded, semester grade point average (SGPA),

cumulative grade point average (CGPA), and general performance analysis. **(LO 5.2)**

5.4 List the steps for downloading Spark and Python libraries for analytics and for creating Hive and PySpark contexts. **(LO 5.3)**

5.5 Compose the SchemaRDD for a table. The table, named `toys_tbl` is as follows: (Product categories, ProductId and Product name are three columns) **(LO 5.3)**

Table of Product Categories, ProductId and Product Name

ProductCategory	ProductId	ProductName
Toy_Airplane	10725	Lost Temple
Toy_Airplane	31047	Propeller Plane
Toy_Airplane	31049	Twin Spin Helicopter
Toy_Train	31054	Blue Express
Toy_Train	10254	Winter Holiday Toy_Train

5.6 Compose the SchemaRDD for grade sheets specified in Practice Exercise 5.3. **(LO 5.3)**

5.7 Write the RDD program steps for calculating SGPA and CGPA of a student program of study from file created in Practice Exercise 5.3. **(LO 5.3)**

5.8 Write the program steps for creating an ETL pipeline for monthly and yearly sales analysis from a table. The table consists of data of `toys_company` manufacturing 1600 different toys and 2000 puzzle product categories, up to 20 product types for each puzzle product of product types 100, 200, 400, 800, 1600, 2400 and 500 pieces. **(LO 5.4)**

5.9 List the steps for reporting using the BIRT tool on analysis of sales of the toy company in above Practice Exercise 5.7. **(LO 5.5)**

5.10 List the steps for viewing charts for analysis of sales of the toy company in above Practice Exercise 5.7. **(LO 5.5)**

1

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC#LanguageManualORC-ORCFfileFormat>

2 <https://www.tecmint.com/install-java-jdk-jre-in-linux/>

3 <https://www.vultr.com/docs/how-to-manually-install-java-8-on-ubuntu-16-04>

4 <https://data-flair.training/blogs/create-rdds-in-apache-spark/>

5 <http://data-flair.training/forums/topic/how-to-create-rdd>

Note:

○○● Level 1 & Level 2 category

○●● Level 3 & Level 4 category

●●● Level 5 & Level 6 category

Chapter 6

Machine Learning Algorithms for Big Data Analytics

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- LO 6.1 Understand metric, feature and category variables, evaluate and estimate the relationships, outliers, variances, probability distribution, and the correlations between the variables in variables, items or entities
- LO 6.2 Apply regression analysis using linear, non-linear and multiple regression models,
K-Nearest Neighbours (KNN) distance measures, and predict the expected results
- LO 6.3 Discover similar items and the similarities using distance measures
- LO 6.4 Apply methods for Frequent-Itemsets Mining (FIM), market-basket model, association-rules mining, Apriori algorithm, and method of evaluation of Candidate Rules. Get knowledge of FIM and association rule applications and find the associations and similarities
- LO 6.5 Apply methods of unsupervised machine-learning for clustering a collection, K-means, determine the number of clusters, and perform

cluster diagnostics

- LO 6.6 Apply methods of supervised machine-learning for classification; K-Nearest Neighbour (KNN) classifier, decision trees and Random-Forest, AdaBoost and other ensemble, Naïve Bayes classifiers, Artificial Neural Networks and SVM-based classifiers
- LO 6.7 Design a recommender system using approaches of Collaborative Filtering (CF), Contents-based Filtering (CBF), Knowledge-based Filtering (KBF) and hybrid approaches for making recommendations
- LO 6.8 Get knowledge of Apache Mahout Architecture, the ML algorithms for clustering, classification, collaborative filtering, and design recommender in Big Data environment

RECALL FROM EARLIER CHAPTERS

The most popular open-source analytics-tools are Apache Spark, Python, R, Apache Pig, and Hive, according to a study (Section 5.1). Spark with multiple languages, Python and Scala shells provide *great ease in programming for complex analytics, machine learning* and other solutions (Section 5.2).

Big Data analysis requires scalable distributed computations. Spark scalable MLlib (machine-learning library) consists of the widely used ML algorithms and utility functions for large datasets. MLlib includes the algorithms for optimization-primitives, regression, collaborative filtering, dimensionality reduction, cluster analysis, classification and recommender.

Uses of Machine Learning (ML) and Artificial Neural Networks (ANN) are in analytics, predictive modeling and decisions. The analytics methods include data mining, pattern mining, clusters analysis and detection of anomalies.

Apache Mahout consists of ML algorithms for Big Data analysis (Section 2.2.3 Figure 2.2).

This Chapter focusses on the ML methods of regression analysis based predictions, finding similarities, FIM, clustering, classifiers, recommenders, and introduces Mahout Architecture and features for the ML applications in Big Data analytics.

6.1 ! INTRODUCTION

Analytics uses the mathematical equations, formulae and models. Analytics also uses the statistics, AI, ML and DL, and predict the behaviour of entities, objects and events. Statistics refers to studying organization, analysis of a collection of data, making interpretations and presentation of analyzed results.

Artificial Intelligence (AI) refers to the science and engineering of making computers perform tasks, which normally require human intelligence. For example, tasks such as predicting future results, visual perception, speech recognition, decision making and natural language processing.

Two concepts in AI, ‘*machine learning*’ and ‘*deep learning*’ provide powerful tools for advanced analytics and predictions.

Google-owned company *Deep Mind* developed an Artificial Intelligence (AI) program called AlphaZero, which played 100 chess games in 24 hours, and defeated Stockfish, the highest-rated chess program by 28 games to 0 with 72 games drawn. This was a historical moment. It became a milestone in the history of AI, ML and DL.

The former world champion, Garry Kasparov, noted that achievement of AlphaZero has history-shaping potential. “The ability of a machine to replicate and surpass centuries of human knowledge, is a world-changing tool”. (Garry Kasparov, “*Deep Thinking – Where Artificial Intelligence Ends and Human Creativity Begins*”, published by the author himself, 2017)

Machine Learning – Definition and Usage Examples

Machine Learning (ML) is a field of computer science based on AI which deals with learning from data in three phases, i.e. *collect, analyze* and *predict*. It does not rely on explicitly programmed instructions.

An ML program learns the behavior of a process. The program uses data generated from various sources for training. Learning from the outcomes from common inputs improves future performance from previous outcomes. Learning applies in many fields of research and industry. Learning from study of data enables efficient and logical decisions for future actions.

Advanced ML techniques use unsupervised, semi-supervised or supervised

learning. Supervised learning uses a known dataset (called *training dataset*). Learning enables creation of a *model program* for evaluating outcomes. The program makes future predictions and leads to knowledge discovery. Supervised learning uses output datasets, which are used to train a machine (program) such that the program leads to the desired outputs. Unsupervised learning does not use output datasets to train a machine.

Deep Learning (DL) refers to structured learning (DSL) or hierarchical learning. DL methods are advanced methods, such as artificial neural networks (ANN) such as artificial neural networks (ANN) or neural nets, deep neural networks, deep belief networks and recurrent neural networks. Learning can be unsupervised, semi-supervised or supervised. Applications of DL and ANN include computer vision, speech recognition, Natural Language Processing (NLP), audio recognition, social network filtering, machine translation, bioinformatics and drug design. DL methods give results comparable to and in some cases superior to human experts.

The present chapter describes the ML methods and introduces Mahout Architecture, features and its ML applications. Section 6.2 describes methods of estimating relationships, outliers, variances, probability distribution, errors and correlations in variables, items and entities. Section 6.3 describes regression analysis using linear, non-linear and multiple-regressor models and KNN distance measures for making predictions. [Regressor means an independent (explanatory) variable in regression equation.]

Section 6.4 describes methods of finding similar items, similarities and filtering of similar items. Section 6.5 describes frequent-itemset mining by collaborative filtering of similar itemsets. Section 6.5 also describes associations and association rules mining. Section 6.6 describes methods of finding the clusters. Section 6.7 describes the classifiers for classifying data in datasets. Section 6.8 introduces recommendation system and collaborative, content, knowledge and hybrid recommendation approaches. Section 6.9 describes Apache Mahout and ML algorithms for Big Datasets.

The following sections use a convention for fonts when denoting an absolute value, mean value, function value, vector element, set member, entity or variable using a character or set of characters, entities or elements.

1. $|u|$ represents absolute value of u , means value without sign. For example, consider $|-3|$ and $|+3|$, the value of both is 3.
2. \bar{x} represents mean, average or expected value of x .
3. $F(y, x)$ represents a function with an expression, which finds value of F from the given values of y and x . $F(y, x)$ values depend on one or more dependent variables as a function of one or more independent variables. For example, F depends y as well as x is $F(y, x) = 1/\sqrt{(y+x)^2 + k^2}$. The F also depends on constant k . Another example is $y = F(x)$, for example, $y = \cos(x)$. $F(x)$ represents a function F , which gives value of y , is a dependent variable. The x is an independent variable.
4. V denotes a vector V ($V_1, V_2 \dots$). V is in bold font. V_1 and V_2 are in text font and are elements 1 and 2 of V . The V consists of number of elements $V_1, V_2 \dots$
5. $\|U\|$ represents length of vector U .
6. S denotes a set S ($A, B, C \dots$). Font S is in French script MT or distinct font for English S . The A, B and C are in text font (no bold), and are the members of S . The members can be vectors or subsets. They, when denoted in bold, represent vector elements.

6.2 • ESTIMATING THE RELATIONSHIPS, OUTLIERS, VARIANCES, PROBABILITY DISTRIBUTIONS AND CORRELATIONS

Methods of studying relationships use variables.
Types of variables used are as follows:

LO 6.1

Metric, feature and category variables, Relationships, outliers, variances, probability distribution, and the correlations between the variables, items, or entities

Independent variables represent directly measurable characteristics. For example, year of sales figure or semester of study. Dependent variables represent the characteristics. For example, profit during successive years or grades awarded in successive semesters. Values of a dependent variable depend on the value of the

independent variable.

Predictor variable is an independent variable, which computes a dependent variable using some equation, function or graph, and does a prediction. For example, predicts sales growth of a car model after five years from given input datasets for the sales, or predicts sentiments about higher sales of particular category of toys next year.

Outcome variable represents the effect of manipulation(s) using a function, equation or experiment. For example, CGPA (Cumulative Grade Points Average) of the student or share of profit to each shareholder in a year using profit as the dependent variable. CGPA of a student computes from the grades awarded in the semesters for which student completes his/her studies. A company declares the share of profit to each shareholder in a year after subtracting requirements of money for future growth from the profit.

Explanatory variable is an independent variable, which explains the behavior of the dependent variable, such as linearity coefficient, non-linear parameters or probabilistic distribution of profit-growth as a function of additional investment in successive years.

Response variable is a dependent variable on which a study, experiment or computation focuses. For example, improvement in profits over the years from the investments made in successive years or improvement in class performance is measured from the extra teaching efforts on individual students of a class.

Feature variable is a variable representing a characteristic. For example, apple feature red, pink, maroon, yellowish, yellowish green and green. Feature variables are generally represented by text characters. Numbers can also represent features. For example, red with 1, orange with 2, yellow with 3, yellowish green 4 and green 5.

Categorical variable is a variable representing a category. For example, car, tractor and truck belong to the same category, i.e., a four-wheeler automobile. Categorical variables are generally represented by text characters.

Independent and dependent variables may exhibit a relation or correlation. The relationships may be linear, nonlinear, positive, negative, direct, inverse, scattered or spread. A data point for dependent variable can be an outlier with no relationship.

Data analysis requires studying relationships graphically, mathematically and statistically, studying the outliers, anomalies, variances, correlations, features, categories and probability distributions using a set of variables, and other characteristics. The relationship involves some quantifiable independent variables and the resulting dependent variable or entity. The following subsections explain methods of estimating the relationships, outliers, variances, correlations and probability distributions between a set of variables.

6.2.1 Relationships—Using Graphs, Scatter Plots and Charts

A relationship between two or more quantitative dependent variables with respect to an independent variable can be well-depicted using graph, scatter plot or chart with data points, shown in distinct shapes. Conventionally, independent variables are on the x-axis, whereas the dependent variables on the y-axis in a graph. A line graph uses a line on an x-y axis to plot a continuous function.

A scatter plot is a plot in which dots or distinct shapes represent values of the dependent variable at the multiple values of the independent variable [Section 10.5]. Whether two variables are related to each other or not, can be derived from statistical analysis using scatter plots.

A data point is (x_i, y_i) when dependent variable value = y_i at the independent variable value = x_i . The

$i = 1, 2 \dots n$ for number of data points = n . The i varies with the position of projection of the point on X-axis. Scatter plot represents data points by dots. The dot can also be a bubble, triangle, circle, cross or vertical bar. Size or colour of dot distinguishes the dependent variables on the same plot.

Another method is quantifying two or more dependent variables by columns of different widths with filled colours, shades or patterns. The width quantifies the dependent variable. The column-position quantifies the independent variable.

Examples of dependent variables are sales of five car models in a year, grades in five courses taken in a semester.

6.2.1.1 Linear and Non-linear Relationships

A linear relationship exists between two variables, say x and y , when a straight line ($y = a_0 + a_1 \cdot x$) can fit on a graph, with at least some reasonable degree of accuracy. The a_1 is the linearity coefficient. For example, a scatter chart can suggest a linear relationship, which means a straight line. Figure 6.1 shows a scatter plot, which fits a linear relationship between the number of students opting for computer courses in years between 2000 and 2017.

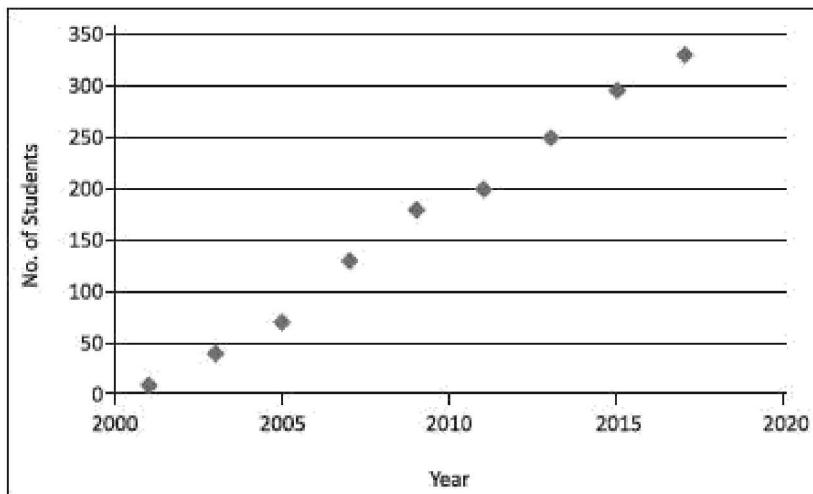


Figure 6.1 Scatter plot for linear relationship between students opting for computer courses in years between 2000 and 2017

A linear relationship can be positive or negative. A positive relationship implies if one variable increases in value, the other also increases in value. A negative relationship, on the other hand, implies when one increases in value, the other decreases in value. Perfect, strong or weak linearship categories depend upon the bonding between the two variables.

A non-linear relationship is said to exist between two quantitative variables when a curve ($y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots$) can be used to fit the data points. The fit should be with at least some reasonable degree of accuracy for the fitted parameters, $a_0, a_1, a_2 \dots$ Expression for y then generally predicts the values of one quantitative variable from the values of the other quantitative variable with considerably more accuracy than a straight line.

Consider an example of non-linear relationship: The side of a square and its area are not linear. In fact, they have quadratic relationship. If the side of a square doubles, then its area increases four times. The relationship predicts the area from the side.

Figure 6.2 shows a scatter plot in case of a non-linear relationship between side of square and its area.

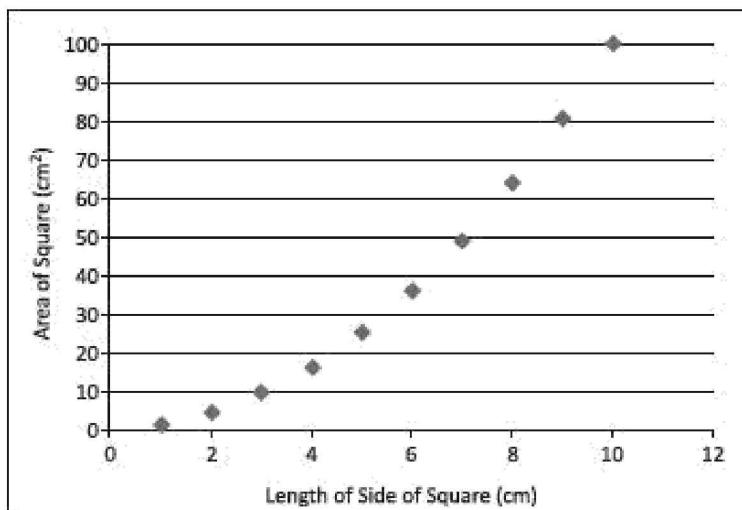


Figure 6.2 Scatter plot in case of a non-linear relationship between side of square and its area

6.2.2 Estimating the Relationships

Estimating the relationships means finding a mathematical expression, which gives the value of the variable according to its relationship with other variables. For example, assume y_m = sales of a car model m in x^{th} year of the start of manufacturing that model. Assume that computations show that the y_m relates by a mathematical expression ($y_m = a_0 + a_1 \cdot x_m + a_2 \cdot x_m^2$) up to an acceptable degree of accuracy, when $a_0 = 490$, $a_1 = 10$ and $a_2 = 5$.

Estimated first year sales, $y_m(1) = (490 + 10) = 500$, second year $y_m(2) = (490 + 10 \times 2 + 5 \times 2^2) = 530$, third year $y_m(3) = (490 + 10 \times 3 + 5 \times 3^2) = 565$, if fit with the desired accuracy, then the results are showing that the expression of y_m estimates the relationship between model m sales in next and other years. The y_m can also predict the sales in 6th or later years. Predictions are up to a certain

degree of certainty.

6.2.3 Outliers

Outliers are data, which appear as they do not belong to the dataset (Section 5.3.3.1). Outliers are data points that are numerically far distant from the rest of the points in a dataset, are termed as outliers. Outliers show significant variations from the rest of the points (Section 1.5.2.2). Identification of outliers is important to improve data quality or to detect an anomaly. The estimating parameters mathematically, statistically, describing an outcome, predicting a dependent variable value, or taking the decisions based on the datasets given for the analysis are sensitive to the outliers.

There are several reasons for the presence of outliers in relationships. Some of these are:

- Anomalous situation
- Presence of a previously unknown fact
- Human error (errors due to data entry or data collection)
- Participants intentionally reporting incorrect data (This is common in self-reported measures and measures that involve sensitive data which participant doesn't want to disclose)
- Sampling error (when an unfitted sample is collected from population).

Population means any group of data, which includes all the data of interest. For example, when analysing 1000 students who gave an examination in a computer course, then the population is 1000. 100 games of chess will represent the population in analysis of 100 games of chess of a grandmaster.

Sample means a subset of the population. Sample represents the population for uses, such as analysis and consists of randomly selected data.

6.2.4 Variance

A random variable is a variable whose possible values are outcomes of a random phenomenon. A random variable is a function that maps the outcomes of unpredictable processes to numerical quantities. A random variable is also called stochastic variable or random quantity. Randomness can be around some

expected mean value or outcome, and with some normal deviation.

Variance measures by the sum of squares of the difference in values of a variable with respect to the expected value. Variance can alternatively be a sum of squares of the difference with respect to value at an origin. Variance indicates how widely data points in a dataset vary. If data points vary greatly from the mean value in a dataset, the variance is large; otherwise, the variance is less. The variance is also a measure of dispersion with respect to the expected value.

A high variance indicates that the data in the dataset is very much spread out over a large area (random dataset), whereas a low variance indicates that the data is very similar in nature.

No variance is sometimes hard to understand in real datasets. The following example illustrates no variance:

EXAMPLE 6.1

Consider an examination where everyone gets the same grades. What does it signify?

SOLUTION

Some measurement problem may have taken place in a situation where either the semester examination questions were so easy that everyone got full marks, or it was so hard that everyone got a zero. Now consider the two types of examinations. After each examination, everyone gets the same score on the test, i.e., everyone gets 'A' grade in one test and everyone gets 'B' in the second test. This is again not telling much

about the study or intelligent quotient of the students. Now, these no variance results signify the extreme case and hard to understand or explain. But in general, differences in scores are always found.

6.2.4.1 Standard Deviation and Standard Error Estimates

The variance is not a standalone statistical parameter. Estimations of other statistical parameters, such as standard deviation and standard error are also used.

Standard Deviation With the help of variance, one can find out the standard

deviation. Standard deviation, denoted by s , is the square root of the variance. The s says, “On an average how far do the data points fall from the mean or expected outcome?” Though the interpretation is the same as variance but s is squared rooted, therefore, less susceptible to the presence of outliers. The formulae for the population and the sample standard deviations are as follows:

$$\text{The Population Standard Deviation: } \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2} \quad (6.1a)$$

$$\text{The Sample Standard Deviation: } \sigma = \sqrt{\frac{1}{S-1} \sum_{i=1}^S (x_i - \bar{x})^2}, \quad (6.1b)$$

where N is number of data points in population, S is number in the sample, μ is expected in the population or average value of x , and \bar{x} is expected x in the sample.

Standard Error The standard error estimate is a measure of the accuracy of predictions from a relationship. Assume the linear relationship in a scatter plot of y (Figure 6.1). The scatter plot line, which fits, is defined as the line that minimizes the sum of squared deviations of prediction (also called the **sum of squares error**). The **standard error of the estimate** is closely related to this quantity and is defined below:

$$\sigma_{\text{est}} = \sqrt{\frac{\sum (y - y')^2}{N}}, \quad \dots (6.2)$$

where s_{est} is the standard error in the estimate, y is an observed value, y' is a predicted value, and N is the number of values observed. The standard error estimate is a measure of the dispersion (or variability) in the predicted values from the expression for relationship. Following are three interpretations from the s_{est} :

1. When s_{est} is small, most of the observed values (y) dots are fairly close to the fitting line in the scatter plot, and better is the estimate based on the equation of the line.
2. When the s_{est} is large, many of the observed values are far away from the line.
3. When the standard error is zero, then no variation exists corresponding to the computed line for predictions. The correlation between the observed

and estimation is perfect.

6.2.5 Probabilistic Distribution of Variables, Items or Entities

Probability is the chance of observing a dependent variable value with respect to some independent variable. Suppose a Grandmaster in chess has won 22 out of 100 games, drawn 78 times, and lost none. Then, probability P of winning P_w is 0.22, P of drawn game P_d is 0.78 and P of losing, $P_l = 0$. The sum of the probabilities is normalized to 1, as only one of the three possibilities exist.

Probability distribution is the distribution of P values as a function of all possible independent values, variables, situations, distances or variables. For example, if P is given by a function $P(x)$, then P varies as x changes. Variations in $P(x)$ with x can be discrete or continuous. The values of P are normalized such that sum of all P values is 1. Assuming distribution is around the expected value \bar{x} , the standard normal distribution formula is:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\bar{x})^2}{2\sigma^2}} \quad (6.3)$$

Normal distribution relates to Gaussian function. Figure 6.3 shows a PDF with normal distribution around $x = \bar{x}$ standard deviation = s and variance = s^2 .

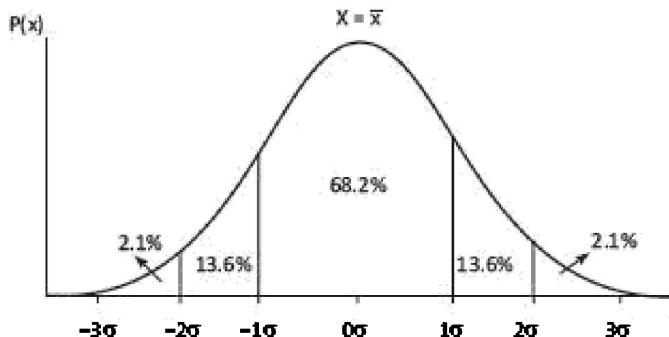


Figure 6.3 Probability distribution function as a function of x assuming normal distribution around $x = \bar{x}$, and standard deviation = s

The figure also shows the percentages of areas in five regions with respect to the total area under the curve for $P(x)$. The variance for probability distribution represents how individual data points relate to each other within a dataset. The

variance is the average of the squared differences between each data value and the mean.

Moments (0, 1, 2 ...) refer to the expected values to the power of (0, 1, 2,) of random variable variance (Section 6.2.5.3). The variance is the second central moment of a distribution, which equals to the square of the standard deviation, and the covariance of the random variable with itself, and it is often represented by s^2 or $\text{var}(x)$. The variance is computed as follows:

$$\sigma^2 = \frac{\sum (x_i - \bar{x})^2}{N} \quad (6.4)$$

Assume that probability distribution (PDF) is normal, called Gaussian distribution, which is like a bell-shaped curve (Figure 6.3). The PDF of the normal distribution is such that 68% of area under the PDF is within $(\bar{x}, + s)$ and $(\bar{x}, - s)$, 95% of area under the PDF is within $(\bar{x} + 2s)$ and $(\bar{x}, - 2s)$ and 99.7% is within $(\bar{x}, + 3s)$ and $(\bar{x}, - 3s)$.

Standard deviation and empirical rule help in computing the population distribution over 68%, 95% and 99.7% of data under normally distributed population. This further helps in forecasting. The following example explains the meaning of population, expected values, normalized probabilities, PDF and interpretation using mean value.

EXAMPLE 6.2

Assume that N students gave the examination. Let N_1 is number of students obtained grade pointer average = 1, N_2 got 2, ..., N_{10} got 10. Highest-grade pointer is 10.0. Grade pointer obtained is not a random variable. Grade pointer variation is a random variable with an expected value and standard deviation.

Expected value among the distributed x_i values, where i varies discretely from 0.0 to 10.0 will depend on the expected performance of the student. If teaching in the class is very good and students prepare for the examination very well, then expected value of GPA is 8.0 for very good performing students and standard deviation found is 1.0.

- (i) What do you mean by population? What do you mean by sample?

- (ii) What will be the normalized probabilities?
- (iii) How will you define Probability Distribution Function (PDF)?
- (iv) How will you interpret the results in terms of normal distribution?
- (v) When will you interpret the results as poor and poorer in terms of normal distribution?

SOLUTION

- (i) Population is GPA of all the students of the university who gave the examination. Population size is N. Sample means datasets used in the analysis. It can be N or less than N students and GPA of each one.
- (ii) Probability that students obtained grade pointer 1 is $\left(\frac{N_1}{N}\right)$, 2 is $\left(\frac{N_2}{N}\right)$, ... on normalization of probability. ($N = N_1 + N_2 + \dots$)
- (iii) PDF represents a curve for independent variable x between $GPA = 0$ and $GPA = 10$, such that the sum of all P values is 1, where P_i is the ratio of number of students getting $GPA = i$ with respect to the total population N or the sample.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\bar{x})^2}{2\sigma^2}} \quad (6.5)$$

between $x = 0$ and 10.0 , where $s = 1.0$ and $x- = 8.0$.

- (iv) GPA value is 8.0 and standard deviation is 1.0, which means 68% of the students will get GPAs between 7.0 and 9.0, 95% between 6.0 and 10.0, and 99.7% between 5.0 and 10.0.
- (v) The expected value of 3.0 (less than 3.0) and standard deviation of 1.0 means poor performance of students because 68% students get between 2.0 and 4.0. The expected value of 3.0- (less than 3.0, say 2.5) and standard deviation of 1.5 means poorer performance of students because 68% students get between 1.0 and 4.0.

6.2.5.1 Kernel Functions

A probability or weight can be represented by a kernel function¹ like a Gaussian or tri-cube function. (Kernel in English means some thing central and key (important) part. For example, the kernel inside a walnut's shell is important because it is the edible part. Kernel in an operating system is key or central component.)

Kernel function is a function which is a central or key part of another function. For example, Gaussian kernel function is the key part of the probability distribution function [Equation (6.5)]. Figure 6.3 shows the probability normal distribution, which is a Gaussian function based on the Gaussian kernel function.

A kernel function¹, $K^*(u)$ defines as

$$K^*(u) = \lambda \cdot K(\lambda \cdot u), \quad (6.6a)$$

where $\lambda > 0$. Gaussian kernel function is

$$K^*(x) = \left[\frac{1}{(\sqrt{2\pi})} \right] e^{-\frac{x^2}{2}}, \quad (6.6b)$$

and when $u = \frac{x - \bar{x}}{\sigma}$, the distribution function is proportional to

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}.$$

$\lambda = \left(\frac{1}{\sigma\sqrt{2}} \right)$ in Equation (6.3).

Tricube kernel function is:

$$K^*(u) = (70/81) (1 - |u|^3)^3 \lambda \cdot K(\lambda \cdot u), \quad (6.6c)$$

where $|u| \leq 1$.

6.2.5.2 Moments

Moments (0, 1, 2, ...) refer to expected values to the powers of (0, 1, 2 ...) of random variable variance. 0th moment is 1, 1st moment = $E(x) = \bar{x}$, (expected value), 2nd moment is squared $V[(x_i - \bar{x})^2] = \text{sum of product of } (x_i - \bar{x})^2$, and $P(x = x_i)$.

Here, P is the probability at $x = x_i$ when i is varying from 1 to n , for n values of random variable x . The r^{th} moment is r^{th} power of variance $V[(x_i - \bar{x})^r]$. Moments are evaluated from the results obtained for the randomly distributed probabilistic values of the variable, such as sales. 1st moment assigns equal weight to variances of outliers and inliers, i.e., equal weight for variance of each. 2nd moment assigns higher weight to outliers compared to inliers. 3rd moment assigns greater weight to outliers compared to inliers. Moment can be defined with respect to the origin, and in that case, $x-$ is considered 0.

Let P is along y axis and variable x on x axis. Central moment means that moments compute taking \bar{x} , equals to variable x at x axis point where the probability curve partitions equally by a vertical axis, parallel to the y axis.

6.2.5.3 Unequal Variance Welch's t-test

A test in statistics is unequal-variance t test, also called Welch t test.

- (i) The test assumes that two groups of data are sampled data which consist of Gaussian distributed populations (Equation (6.3)).
- (ii) The test does not assume those two populations have the same standard deviation.

Unequal variances t -test is a two-sample location test. It tests the hypothesis that two populations have equal means. (*Hypothesis* means making assumption statements about certain characteristics of the population. For example, an assumption that most students of a specific professor will excel as a programmer. Hypothesis when tested for a decade may pass or fail depending up on whether the statistically significant results show that the students of that professor really excelled as programmers.)

Welch's t -test is an adaptation of student's t -test in statistics. The t -test is more reliable when the two samples have unequal variances and unequal sample sizes.

6.2.5.4 Analysis of Variance (ANOVA)

An ANOVA test is a method which finds whether the fitted results are significant or not. This means that the test finds out (infer) whether to reject or accept the null hypothesis. Null hypothesis is a statistical test that means *the hypothesis that "no significant difference exists between the specified populations"*. Any observed

difference is just due to sampling or experimental error.

Consider two specified populations (datasets) consisting of yearly sales data of Tata Zest and Jaguar Land Rover models. The statistical test is for proving that yearly sales of both the models, means increments and decrements of sales are related or not. Null hypothesis starts with the assumption that no significant relation exists in the two sets of data (population).

The analysis (ANOVA) is for disproving or accepting the null hypothesis. The test also finds whether to accept another alternate hypothesis. The test finds that whether testing groups have any difference between them or not.

Analysis of variance (ANOVA) is a useful technique for comparing more than two populations, samples, observations or results of computations. It is used when multiple sample cases are involved. Variation between samples and also within sample items may exist. For example, compare the effect of three different types of teaching methodologies on students. This may be done by comparing the test scores of the three groups of 20 students each. This technique provides inferences about whether the samples have been drawn from populations having the same mean. It is done by examining the amount of variation within each of these samples, relative to the amount of variation between the samples.

F-test F-test requires two estimates of population variance— one based on variance between the samples and the other based on variance within the samples. These two estimates are then compared for F-test:

$$F = \frac{E1(V)}{E2(V)} \quad (6.7)$$

where $E1(V)$ is an estimate of population variance between the two samples and $E2(V)$ is an estimate of population variance within the two samples. Several different F-tables exist. Each one has a different level of significance. Thus, look up the numerator degrees of freedom and the denominator degrees of freedom to find the critical value.

The value of F calculated using the above-mentioned formula is to be compared to the critical value of F for the given degrees of freedom. If the F value calculated is equal or exceeds the critical value, then significant differences between the means of samples exist. This reveals that the samples are not drawn from the same population and thus null hypothesis is rejected.

6.2.5.5 No Relationship Case

Statistical relationship is a dependence or association between two random variables or bivariate data. Bivariate means ‘two variables’. In other words, there are two types of data. Relationships between variables need to be studied and analyzed before drawing conclusions based on it. One cannot determine the right conclusion or association when no relationship between the variables exists.

6.2.6 Correlation

Correlation means analysis which lets us find the association or the absence of the relationship between two variables, x and y . Correlation gives the strength of the relationship between the model and the dependent variable on a convenient 0-100% scale.

R-Square R is a measure of correlation between the predicted values y and the observed values of x . **R-squared** (R^2) is a goodness-of-fit measure in linear-regression model. It is also known as the coefficient of determination. R^2 is the square of R , the coefficient of multiple correlations, and includes additional independent (explanatory) variables in regression equation.

Interpretation of R-squared The larger the R^2 , the better the regression model fits the observations, i.e., the correlation is better. Theoretically, if a model shows 100% variance, then the fitted values are always equal to the observed values, and therefore, all the data points would fall on the fitted regression line.

Correlation differs from a regression analysis. Regression analysis predicts the value of the dependent predictor or response variable based on the known value of the independent variable, assuming a more or less mathematical relationship between two or more variables within the specified variances.

6.2.6.1 Correlation Indicators of Linear Relationships

Correlation is a statistical technique that measures and describes the ‘strength’ and ‘direction’ of the relationship between two variables. Let us explore the relations between only two variables. The significant questions are:

Does y increase or decrease with x ? For example, expenditure increases with income or does the number of patients decrease with proper medication. (Direction)

- (i) Suppose y does increase with x ; then, how fast?
- (ii) Is this relationship strong?
- (iii) Can reliable predictions be made? That is, if one tells the income, can the expenditure be predicted?

Relationships and correlations enable training model on sample data using statistical or ML algorithms. Statistical correlation is measured by the coefficient of correlation. The most common correlation coefficient, called the *Pearson product-moment correlation coefficient*. It measures the strength of the linear association between variables. The correlation r between the two variables x and y is:

$$r = \left[\frac{1}{(n-1)} \right] \times \sum \left\{ \left[\frac{(x_i - \bar{x})}{s_x} \right] \times \left[\frac{(y_i - \bar{y})}{s_y} \right] \right\}, \quad (6.8a)$$

where n is the number of observations in the sample, x_i is the x value for observation i , \bar{x} is the sample mean of x , y_i is the y value for observation i , \bar{y} is the sample mean of y , s_x is the sample standard deviation of x , and s_y is the sample standard deviation of y .

Summation is over all n values of i , $i = 1, 2, \dots, n$.

[r^2 is square of sample correlation coefficient between the observed outcomes and the observed predictor values, and includes intercept on y-axis in case of linear regression.]

Use of Statistical Correlation Assume one sample dataset is $\{u_1, \dots, u_n\}$ containing n values of a parameter r . The $r_{u,i}$ is i -th data point in dataset u . ($i = 1, 2, \dots, n$). Another sample dataset is $\{v_1, \dots, v_n\}$ containing n values of r . $r_{v,i}$ is i -th data point in dataset v . Let the correlation among two samples is being measured. Sample Pearson correlation metric c_r measures how well two sample datasets fit on a straight line.

$$c_r(u, v) = \frac{\sum_i (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2 \sum_i (r_{v,i} - \bar{r}_v)^2}} \quad \dots (6.8b)$$

where the summations are over the values of parameter in the datasets.

Three other similarities based on correlation are:

- (i) Constrained Pearson correlation – It is a variation of Pearson correlation that uses midpoint instead of mean rate.
- (ii) Spearman rank correlation – It is similar to Pearson correlation, except that the ratings are ranks.
- (iii) Kendall's G correlation – It is similar to the Spearman rank correlation, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation.

Numerical value of correlation coefficient ranges from +1.0 to -1.0. It gives an indication of both the strength and direction of the relationship between variables.

In general, a correlation coefficient $r > 0$ indicates a positive relationship; $r < 0$ indicates a negative relationship; $r = 0$ indicates no relationship (or that the variables are independent of each other and not related). Here $r = +1.0$ describes a perfect positive correlation and $r = -1.0$ describes a perfect negative correlation.

The closer the coefficients are to +1.0 and -1.0, the greater is the *strength* of the relationship between the variables.

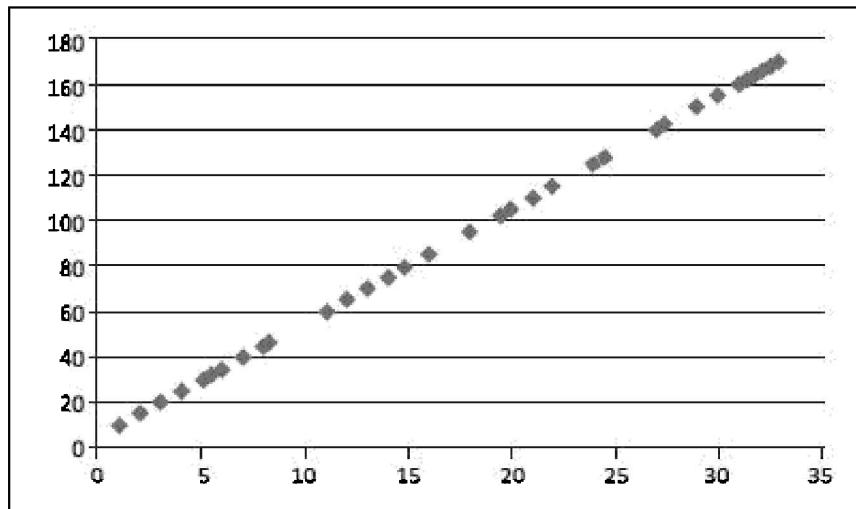
Table 6.1 gives rough guidelines on the strength of the relationship (though many experts would somewhat disagree on the choice of boundaries).

Table 6.1 The strength of the relationship as a function of r

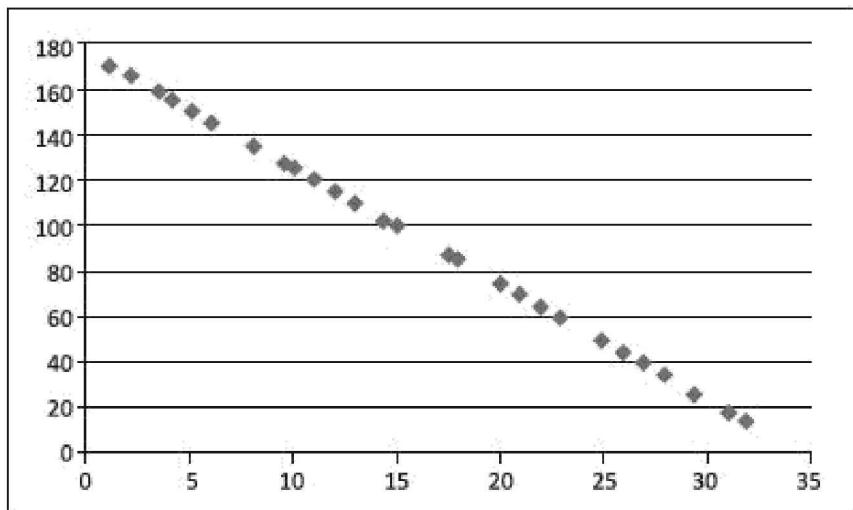
Value of r	Strength of relationship
-1.0 to -0.5 or 1.0 to 0.5	Strong
-0.5 to -0.3 or 0.3 to 0.5	Moderate
-0.3 to -0.1 or 0.1 to 0.3	Weak
-0.1 to 0.1	None or very weak

Correlation is only appropriate for examining the relationship between meaningful quantifiable data (such as, temperature, marks, score) rather than categorical data, such as gender, color etc. Figure 6.4 shows perfect and

imperfect, linear positive and negative relationships, and the strength and direction of the relationship between variables



Perfect Positive Linear
Relationship ($r = 1$)



Perfect Negative Linear
Relationship ($r = -1$)

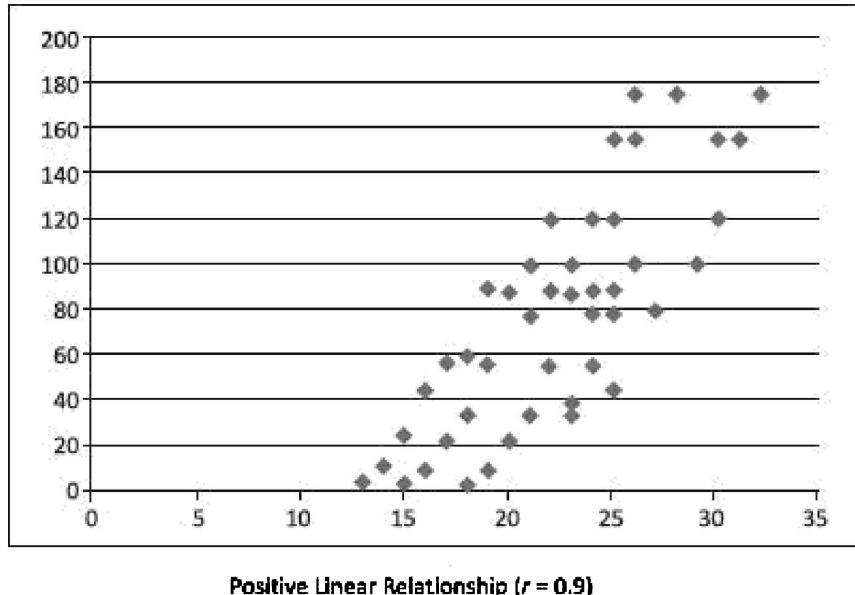
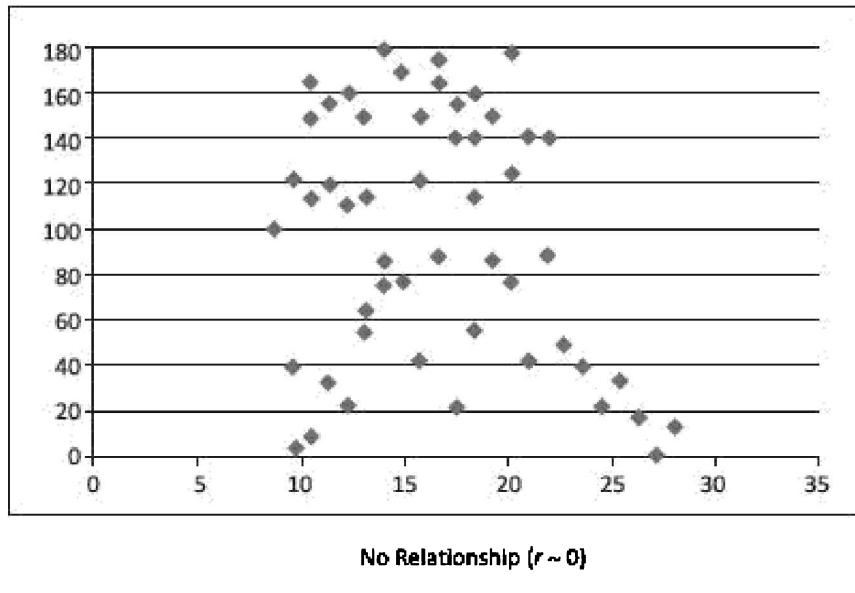


Figure 6.4 Perfect and imperfect, linear positive and negative relationships, and the strength and direction of the relationship between variables

Self-Assessment Exercise linked to LO 6.1

1. Define non-linear relation. Plot on the same graph, a company car sales,

y for its two models every year between 2012 to 2017, using the formula ($y_m = a_0 + a_1 \cdot x_m + a_2 \cdot x_m^2$). How will you predict the sales in 2010? Assume for first model $a_0 = 490$, $a_1 = 10$ and $a_2 = 5$. Assume for second model $a_0 = 4900$, $a_1 = 100$ and $a_2 = 50$. Assume, $x_m = 0$ for year 2011, $x_m = 1$ for 2012 and $x_m = 6$ for 2017.

2. How does the $P(x)$ vary in normal distribution when expected mean is at $x = 6.0$ and standard deviation s is 1.0? Show a plot of $P(x)$ and x and points at deviations of 1.0, 2.0 and 3.0 (means at σ , 2σ and 3σ).
3. Define mean, variance and standard deviation. How do the 0th moment, 1st moment, 2nd moment and 3rd moment compute from the values and their probabilities?
4. When will you perform t-test and F-test?
5. What does variable R-squared mean? How is the correlation parameter between predicted valued and observed value evaluated? When do you use R , r , R^2 and when r^2 ?
6. Consider correlation r between two variables. How do you interpret $r > 0$, $r < 0$ and $r = 0$?
7. How is the inference made that two variables do not correlate?

6.3 | REGRESSION ANALYSIS

Correlation and regression are two analyses based on multivariate distribution. A multivariate distribution means a distribution in multiple variables.

Suppose a company wishes to plan the manufacturing of Jaguar cars for coming years. The company looks at sales data regressively, i.e., data of previous years' sales. Regressive analysis means estimating relationships between variables. Regression analysis is a set of statistical steps, which estimate the relationships among variables. Regression analysis may require

LO 6.2

Regression analysis using linear and non-linear regression models, K-Nearest-Neighbours, and using distance measures for predictions

many techniques for modeling and performing the analysis using multiple variables. The aim of the analysis is to find the relationships between a dependent variable and one or more independent, outcome, predictor or response variables. Regression analysis facilitates prediction of future values of dependent variables.

It helps to find how a dependent variable changes when variation is in an independent variable among a set of them, while the remaining independent variables in the set are kept fixed.

Non-linear regression equation is as follows:

$$y = a_0 + a_1x + a_2x^2 + a_3x^3, \quad (6.9)$$

where number of terms on the right-hand side are 3 or 4. Linear regression means only the first two terms are considered. The following subsections describe regression analysis in detail.

6.3.1 Simple Linear Regression

Linear regression is a simple and widely used algorithm. It is a supervised ML algorithm for predictive analysis. It models a relationship between the independent predictor or explanatory, and the dependent outcome or variable, y using a linearity equation.

$$y = f(a_0, a_1) = a_0 + a_1x, \quad (6.10)$$

where a_0 is a constant and a_1 is the linearity coefficient.

Simple linear regression is performed when the requirement is prediction of values of one variable, with given values of another variable. The following example explains the meaning of linear regression.

EXAMPLE 6.3

How can a university student's GPA be predicted from his/her high school percentage (HSP) of marks?

SOLUTION

Consider a sample of ten students for whom their GPAs and high school scores, HSPs, are known. Assume linear regression. Then,

$$GPA = b_1 \cdot HSP + A$$

... (6.11)

Figure 6.5 shows a simple linear regression plot for the relationship between the college GPA and the percentage of high school marks. Plot the values on a graph, with high school scores in percentage on the x axis and GPA on the y axis.

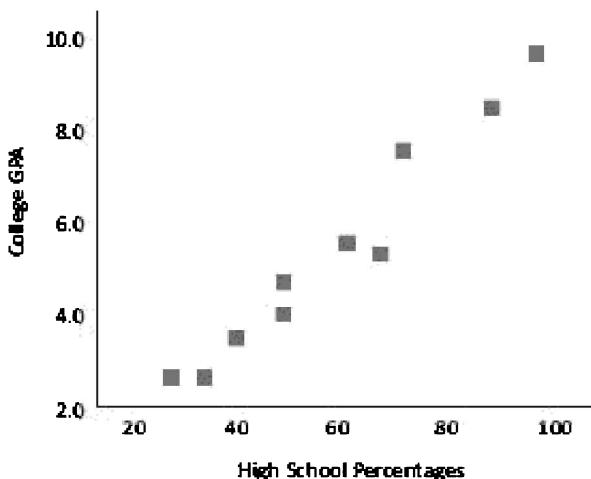


Figure 6.5 Linear regression relationship between college GPA and percentage of high school marks

Whenever a perfect linear relationship between GPA and high school score exists, all 10 points on the graph would fit on a straight line. However, this is never the case. Whenever an imperfect linear relationship exists between these two variables, a cluster of points on the graph, which slope upward, may be obtained. In other words, students who got more marks in high school should get more GPA in college as well.

One variable, denoted by x , is regarded as the predictor, explanatory or independent variable. The other variable, denoted by y , is regarded as the response, outcome or dependent variable.

The purpose of regression analysis is to come up with an equation of a line that fits through a cluster of points with minimal amount of deviation from the line. The best-fitting line is called the *regression line*. The deviation of the points from the line is called an ‘error’. Once this regression equation is obtained, the GPA of a student in college examinations can be predicted provided his/her high

school percentage is given. Simple linear regression is actually the same as a correlation between independent and dependent variables.

Figure 6.6 shows a simple linear regression with two regression lines with different regression equations. Looking at the scatter plot, two lines can fit best to summarize the relation between GPA and high school percentage.

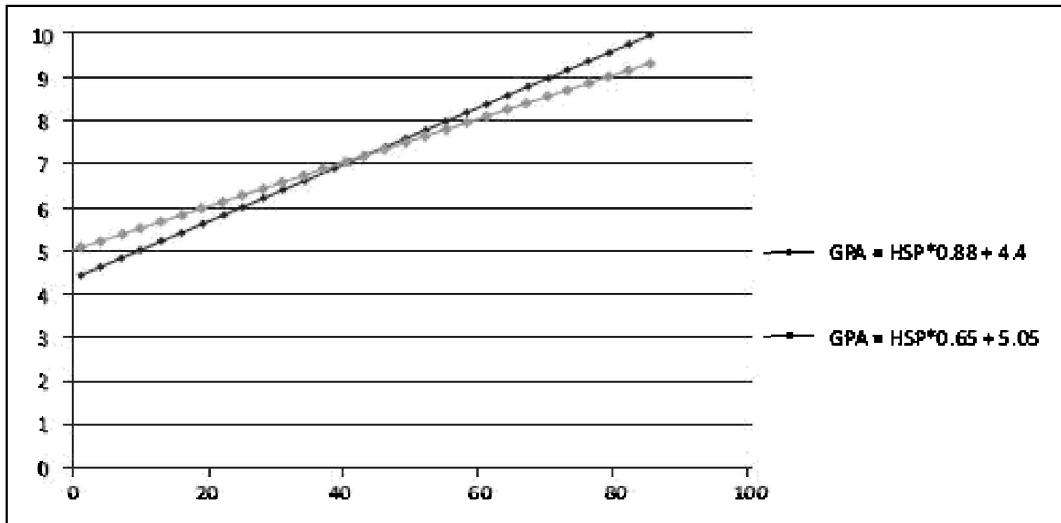


Figure 6.6 Linear regression relationship with two regression lines with different coefficient in regression equation

Following notations can be used for examining which of the two lines is a better fit:

1. y_i denotes the observed response for experimental unit i
2. x_i denotes the predictor value for experimental unit i
3. \hat{y}_i is the predicted response (or fitted value) for experimental unit i

Then, the equation for the best fitting line using a sum of the error estimating function is:

$$\hat{y}'_i = a'_0 + a'_1 x_i \quad (6.12)$$

where a'_0 and a'_1 are the coefficients in Equation (6.10). Use of the above equation to predict the actual response y_i , leads to a prediction error (or residual error) of size:

$$e_i = y_i - \hat{y}_i \quad (6.13)$$

6.3.2 Least Square Estimation

Assume n data-points, $i = 1, 2, \dots, n$. A line out of two lines (Figure 6.6) that fits the data best will be one for which the sum of the squares of the n prediction errors (one for each observed data point) is as small as possible. This is the ‘least squares criterion’, which says that the best fit is one, which ‘minimizes the sum of the squared prediction errors’. This implies that when the equation of the best fitting line is:

$$\hat{y}_i = b_0 + b_1 x_i \quad (6.14)$$

where b_0 and b_1 are the coefficients which minimize the errors. The coefficients values make the sum of the squared prediction errors as small as possible. Thus,

$$\text{Minimize } Q = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (6.15)$$

Q is also called chi-square function. To minimize $Q = \sum_{i=1}^n (y_i - (b_0 + b_1 x_i))^2$, compute the derivative with respect to b_0 and b_1 , set to 0, respectively, and get the ‘least squares estimates’ for b_0 and b_1 as follows:

$$b_0 = \bar{y} - b_1 \bar{x}, \quad (6.16)$$

and

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad \dots (6.17)$$

The derivative of a dependent variable with respect to the independent variable is also called a gradient. Sections 6.7.1.3 and 6.7.3 describe the use of ‘gradient descent’, i.e., a gradient’s descent towards convergence when optimizing for minimum values of gradient descent.

For obtaining the best-fit line here, the sum of the squared prediction error Q is minimized. Since the objective in the regression analysis is to minimize Q , Q is called objective function.

6.3.3 Multiple Regressions

A criterion variable can be predicted from one predictor variable in simple linear regression. The criterion can be predicted by two or more variables in **multiple regressions**. The following example explains the meaning of multiple regression and coefficients.

EXAMPLE 6.4

Recall Example 6.3 where an assumption that university examination GPA depends on past examination HSP was made. Now assume that GPA depends on HSP as well as internal assessment (IA) at the university.

- (i) How will you predict a student GPA on the basis of the HSP and IA during university study?
- (ii) What do the coefficients tell?

SOLUTION

- (i) Regression analysis requirement is to find a linear combination of HSP and IA that best predicts overall GPA. Regression relation gives GPA:

$$\text{GPA} = b_1 \cdot \text{HSP} + b_2 \cdot \text{IA} + b_0 \quad (6.18)$$

where b_0 , b_1 and b_2 are regression coefficients.

- (ii) With multiple independent variables, the coefficients tell how much the dependent (response) variable is expected to increase when the independent (predictor) variable increases by unit value, holding all the other independent variables constant. Remember, the units by which the variables are measured differ for different models. For example, assume $y = 1 + 2x_1 + 3x_2$. When x_2 is constant, for each change of 1 unit in x_1 , y changes 2 units.

Multiple regressions are used when two or more independent factors are involved. These regressions are also widely used to make short- to mid-term predictions to assess which factors to include and which to exclude. Multiple regressions can be used to develop alternate models with different factors.

More than one variable can be used as a predictor with multiple regressions. However, it is always suggested to use a few variables as predictors necessarily, to get a reasonably accurate forecast. The prediction takes the form:

$$y = a + c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (6.19)$$

where a is the intercept of line on the y axis (means value of y when all independent variable values = 0). The c_1 , c_2 , ..., and c_n are coefficients, representing the contributions (weights) of the independent variables x_1 , x_2 , ..., x_n in the calculation of y .

Multiple regression analysis, often referred to simply as regression analysis, examines the effects of multiple independent variables on the value of a dependent variable or outcome.

Statistical significance means that the observer can be confident that the findings are real, and not just a coincidence, for the given data. Regression calculates a coefficient for each independent variable and its statistical significance, to estimate the effect of each independent variable on the dependent variable. An example of a regression study is to examine the effect of education, experience, gender and social background on income.

6.3.4 Modelling Possibilities using Regression

Regressions range from simple models to highly complex equations. Two primary uses for regression are forecasting and optimization. Consider the following examples:

1. Using linear analysis on sales data with monthly sales, a company could forecast sales for future months.
2. For the funds that a company has invested in marketing a particular brand, an analysis of whether the investment has given substantial returns or not can be made.
3. Suppose two promotion campaigns are running on TV and Radio in parallel. A linear regression can confine the individual as well as the combined impact of running these advertisements together.
4. An insurance company exploits a linear regression model to obtain a tentative premium table using predicted claims to Insured Declared Value

ratio.

5. A financial company may be interested in minimizing its risk portfolio and hence want to understand the top five factors or reasons for default by a customer.
6. To predict the characteristics of child based on the characteristics of their parents.
7. A company faces an employment discrimination matter in which a claim that women are being discriminated against in terms of salary is raised.
8. Predicting the prices of houses, considering the locality and builder characteristics in a locality of a particular city.
9. Finding relationships between the structure and the biological activity of compounds through their physical, chemical and physicochemical traits is most commonly performed with regression techniques.
10. To predict compounds with higher bioactivity within groups.

6.3.5 Predictions using Regression Analysis

Regression analysis is a powerful technique used for predicting the unknown value of a variable from the known value of another variable. Regression analysis is generally a statistical method to deal with the formulation of a mathematical model depicting the relationship amongst dependent and independent variables. The dependent variable is used for the purpose of prediction of the values. One or more variables whose values are hypothesized are called independent variables. The prediction for the dependent variable can be made by accurate selection of independent variables to estimate a dependent variable.

Two steps for predicting the dependent variable:

1. *Estimation step:* A function is hypothesized and the parameters of the function are estimated from the data collected on the dependent variable.
2. *Prediction step:* The independent variable values are then input to the parameterized function to generate predictions for the dependent variable.

Consider an example of data that contain two variables, viz., crop yield and rainfall. Assume that the yield depends on rainfall (in certain critical growth phases). Using past yield data as a function of rainfall, the crop yield can be predicted. The application of linear regression upon these two variables will generate a linear equation, $y = a + b.x$, where y and x variables denotes crop yield and rainfall, respectively. Constants, a and b are the model's parameters known as the intercept and slope of the equation.

6.3.6 K-Nearest-Neighbour Regression Analysis

Consider the saying, 'a person is known by the company he/she keeps.' Can a prediction be made using neighbouring data points? K-Nearest Neighbours (KNN) analysis is an ML based technique using the concept, which uses a subset of $K = 1, 2$ or 3 neighbours in place of a complete dataset. The subset is a training dataset.

Assume that population (all data points of interest) consist of k -data points. A data point independent variable is x_i , where $i = 1$ to k . K-Nearest Neighbours (KNN) is an algorithm, which is usually used for classifiers. However, it is useful for regression also. Predictions can use all k examples (global examples) or just K examples (K -neighbours with $K = 1, 2$ or 3). It predicts the unknown value y_p using predictor variable x_p using the available values at the neighbours. The training dataset consists of available values of y_{ni} at x_{ni} with $n_i = 1$ to K , where n_i is the K -the neighbour, means just the local examples.

A subset of training dataset restricts k to K -neighbours, where $K = 1, 2$ or 3 . This means using local values near the predictor variable. $K = 1$ means the nearest neighbour data points. $K = 2$ means the next nearest neighbour data points (x_i, y_i) . $K = 3$ means the next to next nearest neighbour data points (x_i, y_i) .

First find all available neighbouring target (x_i, y_i) cases and then predict the numerical value to be predicted based on a similarity measure. Prediction methods are as follows:

- (i) Simple interpolation, when predictor variable is outside the training subset
- (ii) Extrapolation, when predictor variable is outside the training subset
- (iii) Averaging, local linear regression or local-weighted regression.

KNN analysis assumes that weight is inversely proportional to the square of distance ($w \propto D^{-2}$), inverse of the distance ($w \propto D^{-1}$) or inverse of q^{th} power of the distance ($w \propto D^{-q}$) called Euclidean D_{Eu} , Manhattan D_{Ma} and Minkowski D_{Mi} distances, respectively. When predicting, a weight assignment may require computations using a kernel function¹ like a Gaussian or tri-cube function (Section 6.2.5.1) in cases where the dependent variable varies according to the kernel function.

Assume continuously varying values as a function of independent variables. Assume v denotes the number of variables, independent as well as dependent. The following equations give the KNN distances in v -dimensional space for the purpose of using weights.

Euclidean Distance The following equation computes the Euclidean distance D_{Eu} :

Sum of the squared Euclidean distance, $[D_{\text{Eu}}]^2 = \left[\sum_{i=1}^v (x_i - x'_i)^2 \right]$, and

$$\text{Euclidean distance } D_{\text{Eu}} = \left[\sum_{i=1}^v (x_i - x'_i)^2 \right]^{1/2} \quad (6.20a)$$

Sum is over v dimensions. If one independent and one dependent variable, then $v = 2$. For example, if $v = 2$ and two data points are (x_j, y_j) and (x_{j+1}, y_{j+1}) , then Euclidean distance between the points is as follows:

$$\text{Euclidean distance } D_{\text{Eu}} = [(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2]^{1/2} \quad (6.20b)$$

Euclidean distance for three variables $v = 3$ (two independent variables and one dependent variable case) consists of three terms on the right-hand side in Equation (6.20b).

Manhattan Distance The following equation computes the Manhattan distance D_{Ma} :

$$\text{Manhattan distance} \quad D_{\text{Ma}} = \sum_{i=1}^v |x_i - x'_i| \quad = \quad (6.20c)$$

Manhattan distance for three variables $v = 3$ (two independent variables and one dependent variable case) consists of three terms on the right-hand side in Equation (6.20c).

Comparison between Euclidean and Manhattan Distances Basically,

Euclidean distance is the direct path distance between two data points in v -dimensional metric spaces. Manhattan distance is the staircase path distance between them. Staircase distance means to move to the next point, first move along one metric dimension (say, x axis) from the first point, and then move to the next along another dimension (say, y axis).

When $v = 2$, Euclidean distance is the diagonal distance between the points on an x - y graph. Manhattan distances are faster to calculate as compared to Euclidean distances. Manhattan distances are proportional to Euclidean distances in case of linear regression.

Minkowski Distance The following equation computes the Minkowski distance D_{Mi} :

$$\text{Minkowski distance } D_{Mi} = \left\{ \sum_{i=1}^v [(x_i - x'_i)^q] \right\}^{1/q} \quad (6.20d)$$

Hamming Distance When predictions are on the basis of categorical variables, then use the Hamming distance. It is a measure of the number of instances in which corresponding values are found.

$$\text{Hamming Distance, } D_H = \sum_{i=1}^v |x_i - x'_i|. \quad (6.20e)$$

when $x_i = x'_i$ then $D_H = 0$ and when x_i not equal to x'_i then $D_H = 1$. For example, Hamming distance $D_H = 1$ between 101001 11100 and 111001 11100 because just one substitution is needed, change second bit from 0 to 1 at 10th place from the right to left positioned bits. Hamming distance $D_H = 4$ between 111001 00000 and 011001 11100 because we need four substitutions, change 3rd, 4th, 5th and from 0 to 1 and 11th bit from 1 to 0

An application is in text analytics. Hamming distance $D_H = 3$ between ‘Bank notes’ and ‘Java notes’. The distance = 3 because the required number of changes is 3 at B, n and k among two strings. Another application of Hamming distance is in counting the number of data points off from the regression curve (Refer Section 6.4.4.6). Another application is in counting the wrong or distinct characters when comparing two document sentences.

Normalization Concept Normalization factor in p -norm form in a v -dimensional space is

$$x_i = N^{-1} \cdot x_i \text{ where } N = \left(\sum_{i=1}^v |x_i|^p \right)^{1/p} \quad (6.21)$$

Here, x_i is i^{th} component of the vector \mathbf{X} . The total number of components are v . Two-dimensional space $v = 2$, three-dimensional $v = 3$.

The following example explains the meaning of distances, use of Euclidean and Manhattan distances, use distances for predictions, and the KNN regression analysis.

EXAMPLE 6.5

Assume dataset S with two subsets of sets J_{spi} and Z_{spi} for sales and sales percent increase (SPI) for Jaguar Land Rover and Zest models of Tata Motors Company. Assume S is training dataset and consists of data points as per the following table.

Table 6.2 An example of two car models, Jaguar, and Zest (JLRS and ZS), sales and sales percent increase (SPI) in years between 2012 and 2018

Year Y	Number of years from the base year 2012 Y_b	Car model Jaguar sales, JLRS	Car model M, SPI over previous year (M, J_{spi})	Car model Zest sales, ZS	Car model SPI over previous year (M, Z_{spi})
2012	0	1000	(0, 5)	10000	(1, 3)
2013	1	1040	(0, 4)	10400	(1, 4)
2014	2	1123	(0, 8)	11232	(1, 8)
2015	3	1224	(0, 9)	12242	(1, 9)
2016	4	1298	(0, 6)	12977	(1, 6)
2017	5	1428	(0, 10)	13496	(1, 4)
2018	6	1541	(0, 8)	14576	(1, 8)

$M = 0$ means Jaguar Land Rover, and $M = 1$ means Zest.

- (i) Draw two plots, one with scatter set of points with Y and ZS , which means columns 1 and 5 data, second plot between Y_b and J_{spi} with columns 2 and 4.
- (ii) Find the Euclidean 2-NN distance between third and first row data points (2014, 11232) and (2012, 10000).
- (iii) What is the Manhattan 2-NN distance between the third and first row data points (2014, 11232) and (2012, 10000)?

- (iv) What are seven Hamming distance terms of Equation (6.20e) between fourth and six column vectors J_{spi} and Z_{spi} ? Interpret the summed D_{Ha} .
- (v) Assume data point for JLRS as missing for 2015. How do you predict car sales in 2015 assuming missing row for 2015? Use Euclidean distances using 1-NN. How do the results differ when using 2-NN and 3-NN?
- (vi) How do you predict car sales for 2011? Use Euclidean distances.
- (vii) How will you use 1-NN, 2-NN and 3-NN for estimation regression coefficient?
- (viii) How will you calculate Euclidean distances D_{Eu} between values J_{spi} in columns 4 for $Y_b = 3$ and 5?
- (ix) How will you calculate D_{Eu} between value for column 2 $Y_b = 0$ and column 2 Z_{spi} value in column 6 for $Y_b = 1, 3$ and 4?

SOLUTION

- (i) Figure 6.7 shows scatter set of points one for Y and Z_S , which means data points in columns 1 and 5, and second for Y_b and J_{spi} , which means data points in columns 2 and 4.

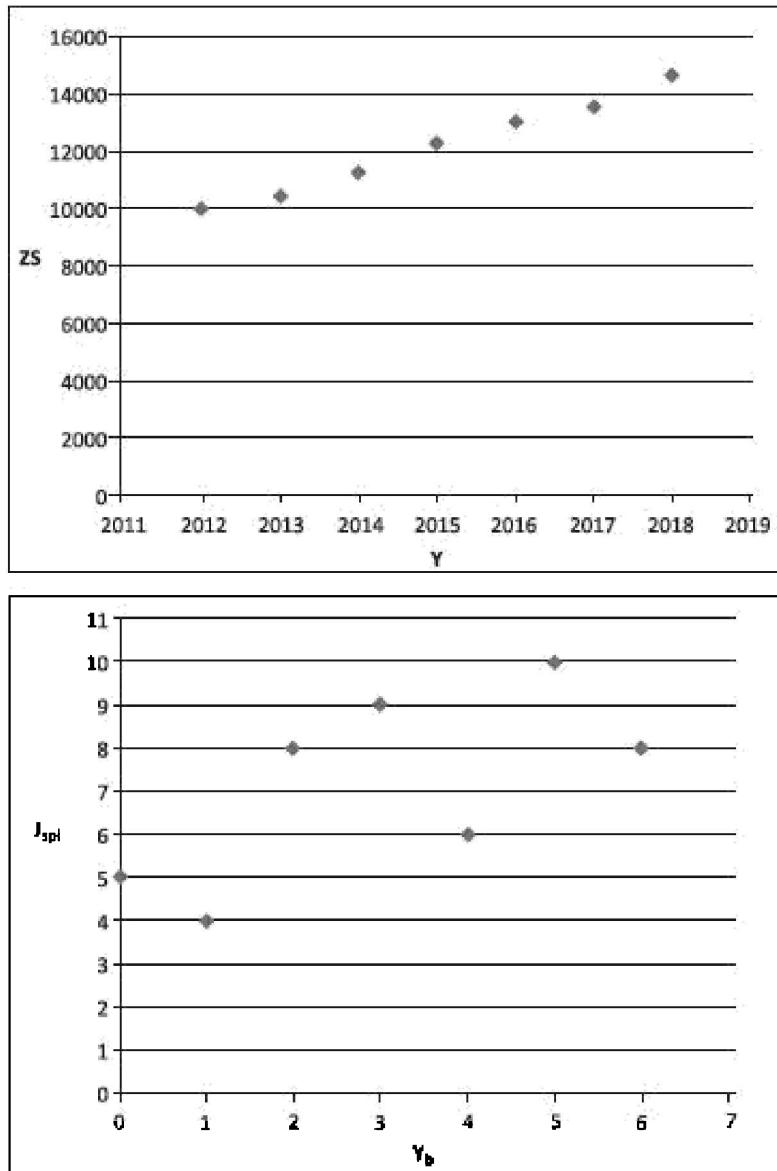


Figure 6.7 Scatter plots for two set of data points, one between Y and ZS, and second between Y_b and J_{spi}

- (ii) Using the equation, $D_{Eu} = [(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2]^{\frac{1}{2}}$, find the Euclidean 2-NN distance between third and first row data points (2014, 11232) and (2012, 10000) $D_{Eu} = [(2014 - 2012)^2 + (11232 - 10000)^2]^{\frac{1}{2}} = [(2)^2 + (1232)^2]^{\frac{1}{2}} = 1232.001$.

- (iii) Using the equation, $D_{Ma} = [(x_j - x_{j+1}) + (y_j - y_{j+1})]$. Manhattan 2-NN distance between third and first row data points (2014, 11232) and (2012, 10000) = $[(2014 - 2012) + (11232 - 10000)] = [2 + 1232] = 1234$.
- (iv) Hamming distances need to compute between fourth and six column vectors Jspi and Zspi are {1, 0, 0, 0, 0, 1, 0} because only in these the Jspi and Zspi differ. That also means that in two years out of seven, increase in sales percentage differs for Jaguar Land Rover and Zest models.
- (v) Lets JLRS missing for 2015 (independent or predictor variable). Since 2014 and 2016 are its 1-NN. Let us choose 1-NN of year 2014, that is 2013. $D_{Eu}(2014, 2013) = \sqrt{(2014 - 2013)^2 + (1123 - 1040)^2} = 83$. Predicted JLRS (2015) = $1123 + 83 = 1246$ by extrapolation, assuming $D_{Eu}(2014, 2013) = D_{Eu}(2014, 2015)$. (weight factors 1)

Years 2012 and 2016 are 2-NNs of 2014. Let us consider $D_{Eu}(2014, 2016) = 175$. Thus, the predicted JLRS(2015) = $(1298 - 175/2) = 1210$ using interpolation (weight factor = 1 per year change).

Similar computations can be made for $D_{Eu}(2014, 2017)$ as 3-NN of 2014 is 2017. D_{Eu} 3-NN = 305. Predicted JLRS(2015) = $(1123 + 305/3) = 1225$.

- (vi) Predicting the car sales for 2011 is an example of extrapolation, when predictor variable is outside the training subset. JLR(2012) is closest point. $D_{Eu}(2012, 2013) = 40$. Predicted JLRS(2011) = $1000 - 40 = 960$.
- (vii) K-NN algorithm is used for estimating regression coefficient. For example, use a weighted average of the k-nearest neighbours, weighted by the inverse of their distance. Compute the Euclidean from the query example to the labeled examples.
 1. Order the labeled examples by increasing distance.
 2. Find a heuristically optimal number k of nearest neighbours.
 3. Calculate an inverse distance weighted average with the k -nearest multivariate neighbours.
- (viii) Euclidean distances between values Jspi_n in columns 4 for Yb = 3 and 5

$$D_{Eu} = \left[(Y_{b_3} - Y_{b_5})^2 + (J_{spi_2} - J_{spi_5})^2 \right]^{1/2}$$

$$= [(3 - 5)^2 + (9 - 10)^2]^{1/2} = [(-2)^2 + (-1)^2]^{1/2}$$

$$D_{Eu} = [5]^{1/2} = 2.236$$

- (ix) D_{Eu} between its value for column 2 $Y_b = 0$ and value of Z_{spi} in column 6 for $Y_b = 1, 3$, and 4

$$D_{Eu} = \left[(Y_{b_0} - Y_{b_1})^2 + (Z_{spi_0} - Z_{spi_1})^2 \right]^{1/2}$$

$$= [(0 - 1)^2 + (3 - 4)^2]^{1/2} = [(-1)^2 + (-1)^2]^{1/2} = 1.414$$

$$D_{Eu} = \left[(Y_{b_0} - Y_{b_3})^2 + (Z_{spi_0} - Z_{spi_3})^2 \right]^{1/2}$$

$$= [(0 - 3)^2 + (3 - 9)^2]^{1/2} = [(-3)^2 + (-6)^2]^{1/2} = 6.708$$

$$D_{Eu} = \left[(Y_{b_0} - Y_{b_4})^2 + (Z_{spi_0} - Z_{spi_4})^2 \right]^{1/2}$$

$$= [(0 - 4)^2 + (3 - 6)^2]^{1/2} = [(-4)^2 + (-3)^2]^{1/2} = 5.0$$

Self-Assessment Exercise linked to LO 6.2

- How does regression analysis predict the value of the dependent variable in case of linear regression?
- (i) Define objective function for least square fitting of coefficients in regression equation.
(ii) How are the best-fitting regression coefficients evaluated?
- When are multiple regressions used? How do multiple regressions predict intermediate term? How do multiple regressions assess which factors to include and which to exclude? How do multiple regressions help in developing alternate models with different factors?
- How is KNN regression used for predicting, considering two variables and $K = 3$? Use training dataset given in Example 6.5.
- How do KNN regression computations differ when using Euclidean and Manhattan distances? Consider two variables and $K = 3$. Use the training dataset given in Example 6.5.

6.4 • FINDING SIMILAR ITEMS, SIMILARITY OF SETS AND COLLABORATIVE FILTERING

Similar item search refers to a data mining method which helps in discovering items which have similarities in datasets. (*Data mining* means discovering previously unknown interesting patterns and knowledge from apparently unstructured data. The process of data mining uses the ML algorithms. Data mining enables analysis, categorization and summarization of data and relationships among data.)

LO 6.3

Finding similar items, applications of Near-Neighbour search; Jaccard similarity of sets; similarity of documents; collaborative filtering as a similar-set problem; and Euclidean, Jaccard, Cosine, Edit and Hamming distances

The following subsections describe methods of finding similar items using similarities, application of near-neighbour search, Jaccard similarity of sets, similarity of documents, Collaborative Filtering (CF) as a similar-set problem, and the distance measures for finding similarities.

6.4.1 Finding Similar Items

An analysis requires many times to find similar items. For example, finding similar excellent performance of students in Python programming, similar showrooms of a specific car model which show high sales per month, recommending books on similar topic such as in Internet of Things by Raj Kamal from McGraw-Hill Higher Education, etc.

6.4.1.1 Application of Near Neighbour Search

Similar items can be found using Nearest Neighbour Search (NNS). The search finds that a point in a given set is most similar (closest) to a given point. A dissimilarity function having larger value means less similar. The dissimilarity function is used to find similar items.

NNS algorithm is as follows: Consider set S having points in a space M . Consider a queried point $q \in M$, which means q is member of M . k-NNS algorithm finds the k -closest (1-NN) points to q in S .

Three problems with the Pearson similarities (6.2.6.1):

1. Do not consider the number of items in which two users' preferences overlap. (e.g., 2 overlap items ==> 1, more items may not be better.)
2. If two users overlap on only one item, no correlation can be computed.
3. The correlation is undefined if series of preference values are identical.

Greater distance means greater dissimilarity. Dissimilarity coefficient relates to a distance metric in metrics space in v -dimensional space. An algorithm computes Euclidean, Manhattan and Minkowski distances using Equations (6.20a) to (6.20d).

Distance metric is symmetric and follows triangular inequality. Meaning of triangular inequality can be understood by an example. Consider three vectors of lengths x , y , and z . Then, triangular inequality means

$z < x + y$. It is similar to the theorem of inequality that the third side of a triangle is less than the sum of two other sides, and never equal. The theorem applies to v -dimensional space also. Dissimilarity can be asymmetric, i.e., triangular inequality is not true (Bergman divergence).

Consider a linear search (also referred as Naïve search) algorithm, Naïve, one of meaning is *simple* in English. Search requires computations of distances to every other point. The algorithm running time is large. The time function, $O(v.c)$ which measures the efficiency of the search algorithm in terms of means $v.c$. The v is dimensionality of M and c is cardinality of S . Cardinality refers to the number of relationships. For example, one independent variable and two dependent variables in a relationship, then cardinality is 3. Cardinality in the context of databases means the uniqueness of values contained in a column fields.

Note: Space partitioning followed by the search algorithm is an efficient method using a k-d tree or R-tree data structure. Search is made after arranging the tree-like data structure. Space partitioning problems become complex in case of high dimensionality.

Naïve search algorithm outperforms space partitioning approaches when using high dimensional spaces M and high cardinality.²

The following example explains the NNS approach to find similar items.

EXAMPLE 6.6

Assume a set S consists of data of a large number of students. The dataset consists of grade points (GPs) in each of the five subjects of study in a semester. The total dataset is for six semesters. Each semester examination awards SGPA_i (Semester Grade Point averages). CGPA_i (Cumulative GPA of ith semester) calculates after end of ith semester after adding the SGPA_is of previous semesters. Assume that each student GP is on a 10-point scale. A student performance in a subject is high (H) if GP is 8.0 or close within ± 1.0 . A student performance in a subject is excellent (E) if GP is 9.0 or close by within ± 1.0 .

- (i) How will you choose independent and dependent variables? What does metric space mean?
- (ii) How will you define a metric space for finding similar performances in a specific subject? How will you define a metric space M for finding similar performance from SGPA_is of the first semester? How will you define a metric space for finding similar performances from CGPA_is of a semester?
- (iii) What will you consider S for finding similarities by NNS?
- (iv) What does nearest neighbour search mean when search is for students with similar excellent performance?
- (v) How will you find students of similar excellent performance by the GPs of a subject, say Java Programming in the second semester?
- (vi) How will you find similar excellent performances by the CGPA?
- (vii) How will you find similar high performances by the SGPA?
- (viii) How will you compute Euclidian and Manhattan distances with respect to query point GP = 8.0 ± 1.0 ? How will you compute dissimilarity?
- (ix) What do you mean by dimensionality of M ? What do you mean by cardinality of S ?

SOLUTION

- (i) Independent variables are student ID, year of study, semester period, name and type (theory or practical) of five subjects. Dependent variables are GP, GPA, SGPA and CGPA. Metric space means a space in which variables are quantifiable. For example, GP, GPA, SGPA and CGPA.
- (ii) Metric space for finding similar performances in a specific subject, Metric space M for finding similar performance in SGPA of first semester, Metric space for finding similar performances from CGPA of a semester:
- (iii) Members of set S are input vectors, each having elements {studentID, CGPA [or SGPA, GPA, T_GPA (GPA of theory subject), P_GPA (GPA of practical subject)]} for each student for finding the similarities by NNS using three distances D1, D2, D3 of first, second and third nearest neighbours.
- (iv) Nearest neighbour search for students with similar excellent performance means search of studentIDs awarded CGPA within the distance 1.0 from 9.0.
- (v) Students of similar excellent performance by the GPs of a subject, say Java Programming, in the second semester means Student IDs with GPs in Java programming within the distance ± 1.0 from 9.0.
- (vi) Similar excellent performance by the CGPA means similar performance of students_IDs with CGPA within the distance ± 1.0 from 9.0.
- (vii) Similar high performance by the SGPA means similar performance of students_IDs with SGPA within the distance ± 1.0 from 8.0.
- (viii) Computation of Euclidian distances with respect query point GP = 8.0 ± 1.0
Computation of Manhattan distances with respect query point GP = 8.0 ± 1.0
- (ix) Dimensionality of M equals the number of independent and dependent variables in metric space for which distances are quantifiable.

Cardinality of S means number of relationships, number of independent but unrelated and dependent unrelated variables. For example, subject_name and subject_type is related to each other. Therefore, subject_name and subject_type are counted as one variable when computing cardinality.

6.4.2 Jaccard Similarity of Sets

Let A and B be two sets. Jaccard similarity coefficient of two sets measures using notations in set theory as shown below:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (6.22)$$

$A \cap B$ means the number of elements or items that are same in sets A and B. $A \cup B$ means the number of elements or items present in union of both the sets. Assume two set of students in two computer courses, Computer Applications CA, and Computer Science CS in a semester. Set CA 40 students opted for Java out of 60 students. Set CS 30 students opted for Java out of 50 students. Jaccard similarity coefficient $J_{\text{java}} (CA, CS) = 30/(60 + 50) \times 100\% = 27\%$. Two sets are sharing 27% of the members for Java course.

(\cap is symbol for intersection in set theory. \cup is symbol for union in set theory.)

6.4.2.1 Similarity of Documents

An application of Jaccard similarity coefficient is in Natural Language Processing (NLP) and text processing. It quantifies the similarity in documents. Computational steps are as follows:

1. Find Bag of Words (Section 9.2.1.4) and remove words such as is, are, does, at, in,
2. Assign weighting factor is the Term frequency and Inverse Document Frequency (TF-IDF). Consider the frequency of words in the document.
3. Find k-shingles. A shingle is a word of fixed length. The k-shingles are the number of times the similar shingles extracted from a document or text. Examples of a shingle are Java, GP, 8.0, Python, 80%, Programming.
4. Find n-grams. A gram is a contiguous sequence of fixed length item (word

or set of characters, letters, words in pairs, triplets, quadruplets, ...) in a document or text. The n-grams are the number of times the similar items (1-grams, 2-grams, ..) extracted from a document or text. The 3-gram examples are Java GP 8.0, Python Programming 7.8, Big Data Analytics, 23A 240C 8LP, the numbers of which are extracted from the text.

5. Compute Jaccard similarity coefficient using Equation (6.22) between the documents.

A number of other methods exist for computing similarity of documents. One method is Latent Semantic Indexing method (LSI). The computational steps are as follows:

- Steps 1 and 2 are the same as above.
- Consider documents into word space. Reduce dimensionality of the projection space. An algebraic model is one that represents text documents as vectors or identifiers, such as how many times a word is present in a document, the index terms or deploy singular value decomposition method.
- Use Cosine Similarity measure between the documents.

Refer Section 9.2 for details on text analysis.

6.4.3 Collaborative Filtering as a Similar-Sets Finding Problem

An analysis requires finding similar sets using collaborative filtering. Collaborative filtering refers to a filtering algorithm, which filters the items sets that have similarities with different items in a dataset.

CF finds the sets with items having the same or close similarity coefficients. Following are some examples of applications of CF:

- Find those sets of students in computer application, and computer science who opt for the Java Programming subject in a semester.
- Find sets of students in Java Programming subjects to whom same teacher taught and they showed excellent performance.

An algorithm finds the similarities between the sets for the CF. Applications of

CF are in many ML methods, such as association rule mining, classifiers, and recommenders.

6.4.4 Distance Measures for Finding Similar Items or Users

Distance measures compute the dissimilarities. Complement of dissimilarity gives similarity. The following subsections describe the distance measures.

6.4.4.1 Definition of a Distance

Distance can be defined in a number of ways. Distance is the measure of length of a line between two values in a two-dimensional map or graph. Set of Equations (6.20) measures distances.

For example, distance between (2014, 6%) and (2018, 8%) on a scatter plot when year is on the x axis and profit % on the y axis is $\text{Distance} = \sqrt{[(2014 - 2018)^2 + (6 - 8)^2]} = \sqrt{(16 + 4)} = 4.47$, using Equation (6.20b). Distance can also be similarly defined in v-dimensional space using Equation (6.20a).

Distances between all members in a set of points can be computed in metrics space using a mathematical equation. Metrics space means measurable or quantifiable space. For example, profit and year on a scatter plot are in metric space of two dimensions. Probability distribution function values are in metric space.

Consider student-performance measures ‘very good’ and ‘excellent’. These parameters are in non-metric space. How are they made measurable? They become measurable when very good is specified as grade point average 8.5 which implies that a score between 8.0 to 9.0 is very good, and define 9.5 which implies that a score between 9.0 to 10.0 is excellent on a 10-point scale.

Consider a chart between number of students passing in examination with best grades vs languages C++, Java, Node.js and Python. Languages are in non-metric space. They become measurable when numbers, say 0, 1, 2 and 3 are assigned for a language for the purpose of using distance measure for similarity analysis.

Distance can be defined as the reciprocal of weight in v-dimensional space. For example, a point at unit distance can be taken as weight $w = 1$, and a point at distance = 2, $w = \frac{1}{2}$ and so on.

Distance can also be defined as dissimilarity coefficient in v-dimensional

space. Greater distance means greater dissimilarity. Subtracting dissimilarity coefficient from 1 gives similarity coefficient. Many different algorithms exist to compute distance and thus similarity between entities, number of users or items. An algorithm computes the distances D_{Eu} , D_{Ma} , D_{Mi} , D_{Ha} [Equations (6.20a to e)] or any other distance metric, for example, Jaccard distance D_{Ja} , cosine distance D_{Cos} , edit distance D_{Ed} .

Jaccard similarity, Cosine similarity, edit distance or correlation methods are used to find out similarities between users.

6.4.4.2 Euclidean Distance

Euclidean distance $D_{Eu} = \left[\sum_{i=1}^n (x_i - x'_i)^2 \right]^{1/2}$, refer Equation (6.20a) in Section 6.3.6 for details.)

6.4.4.3 Jaccard Distance

Equation (6.22) gives $J(A, B)$. Jaccard distance, $D_{Ja}(A, B)$ measures the dissimilarity between two sets. It is equal to result of subtraction of Jaccard similarity coefficient $J(A, B)$ from 1.

$$D_{Ja}(A, B) = 1 - J(A, B) \quad (6.23)$$

(Refer Section 6.4.2 for details.)

6.4.4.4 Cosine Distance

Cosine similarity is a measure of similarity in the inner-product space between two vectors of finite magnitudes. Cosine distance D_{Cos} is measure of dissimilarity between vectors. A measure of cosine distance is in terms of the angle between the vectors. Cosine similarity has low complexity. Cosine distance has applications in text mining, finding similarity of documents, and similarities in sparse vectors, column-vectors (fields) and matrices (Section 3.3.3.1).

Let \mathbf{U} and \mathbf{V} be two non-zero vectors, two documents in the vector space.

$$D_{Cos}(\mathbf{U}, \mathbf{V}) = \frac{\sum_i U_i V_i}{\sqrt{\sum_i U_i^2} \sqrt{\sum_i V_i^2}}, \quad (6.23a)$$

where U_i and V_i are components of \mathbf{U} and \mathbf{V} , respectively, and summation in numerator is over $i = 1$ to N , where N is the number of elements of the vectors.

No Triangular Inequality Property Cosine distances do not exhibit triangular

inequality property, while the Euclidean distances exhibit triangular inequality (Section 6.4.1.1).

Vector Cosine-Based Similarity Vector cosine similarity in terms of angle between two vectors \mathbf{U} and \mathbf{V} is given by equation:

$$\phi_{UV} = \cos^{-1} (\mathbf{U}, \mathbf{V}) = \frac{\mathbf{U} \cdot \mathbf{V}}{\|\mathbf{U}\| \|\mathbf{V}\|} \quad (6.23b)$$

Consider Example 6.5. Let each model have Sales Percentage Increase (SPI) values in successive years. The similarity between SPIs of two models, M_1 and M_2 , is measured by treating each model as a vector of SPIs and computing the cosine of the angle formed by the SPI vectors.

Formally, if \mathbf{P} is $m \times n$ SPI matrix for a model M , then the similarity between two models, M_i and M_j is defined as the cosine in the n -dimensional vectors space corresponding to the i^{th} and j^{th} columns of \mathbf{P} .

The following example illustrates computing of cosine and Euclidean similarities to find similar items.

EXAMPLE 6.7

Consider members of a dataset S in five-dimensional metric space. Assume S subsets are JLR and Z . Data subset members consist of values in two column vectors J_{spi} and Z_{spi} of the elements SPIs. Data subset JLR consists of percentage increase in sales number in a year of Tata Jaguar Land Rovers cars, and Z consists of Zest cars SPIs. (Example 6.5) Assume dataset S consists of data points as per Table 6.2.

- (i) Represent members of dataset S of table data points in five-dimensional metric space consisting of three independent variables Y , Y_b , M and two dependent variables J_{spi} and Z_{spi} .
- (ii) Represent the members of six subsets for values in columns 1, 2, 3 and 5 as elements of vectors Y , Y_b , and matrices (M, J_{spi}) and (M, Z_{spi}) , respectively.
- (iii) Represent the table data points in two-dimensional metric spaces (Y_b, J_{spi}) and (Y_b, Z_{spi}) .

- (iv) Represent the table data points in three-dimensional metric space (Y_b , J_{spi} , Z_{spi}).
- (v) How will you calculate the cosine distance, cosine similarity and angle between the vectors J_{spi} and Z_{spi} ?
- (vi) How will you calculate Euclidean similarity using six neighbour distances D_{Eu} starting from Z_{spi} for $Y_b = 0$ and Z_{spi} values in columns 6 for $Y_b = 1$ to 5?

SOLUTION

- (i) $S\{2012, 0, (0,5), (1,3)\}, \{2013, 1, (0,4), (1,4)\}, \{2014, 2, (0,8), (1,8)\}, \{2015, 3, (0,9), (1,9)\}, \{2016, 4, (0,6), (1,6)\}, \{2017, 5, (0,10), (1,10)\}, \{2018, 6, (0,8), (1,8)\}$
- (ii) $Y = \{2012, 2013, 2014, 2015, 2016, 2017, 2018\}$, $Y_b = \{0, 1, 2, 3, 4, 5, 6\}$, $(M, J_{spi}) = \{(0,5), (0, 4), (0, 8), (0, 9), (0, 6), (0, 10), (0, 8)\}$, and $(M, Z_{spi}) = \{(1, 3), (1, 4), (1, 8), (1, 9), (1, 6), (1, 4), (1, 7)\}$
- (iii) $(Y_b, J_{spi}) = \{(0,5), (0,4), (0,8), (0,9), (0,6), (0,10), (0,8)\}$ and $(Y_b, Z_{spi}) = \{(0,3), (0,4), (0,8), (0,9), (0,6), (0,4), (0,8)\}$
- (iv) $(Y_b, J_{spi}, Z_{spi}) = \{(0,5, 3), (1,4, 4), (2,8, 8), (3,9, 9), (4,6, 6), (5,10, 4), (6,8, 8)\}$.
- (v) $D_{cos}(J_{spi}, Z_{spi}) = \{(5 \times 3) + (4 \times 4) + (8 \times 8) + (9 \times 9) + (6 \times 6) + (10 \times 4) + (8 \times 8)\} / \sqrt{5^2 + 4^2 + 8^2 + 9^2 + 6^2 + 10^2 + 8^2} \times \sqrt{3^2 + 4^2 + 8^2 + 9^2 + 6^2 + 4^2 + 8^2} = 0.951$
 $\text{Cosine similarity} = 1 - D_{cos}(J_{spi}, Z_{spi}) = 1 - 0.951 = 0.049$
 $\text{Angle between } J_{spi}, Z_{spi} = \cos^{-1}(D_{cos}) = 87.191$
- (vi) Use Equation (6.20b) for the computations.
 1. $D_{Eu}(Y_b = 0, Y_b = 1) = \sqrt{(0 - 1)^2 + (3 - 4)^2}$;
 2. $D_{Eu}(Y_b = 1, Y_b = 2) = \sqrt{(1 - 2)^2 + (4 - 8)^2}$;
 3. $D_{Eu}(Y_b = 2, Y_b = 3) = \sqrt{(2 - 3)^2 + (8 - 9)^2}$

$$4. D_{\text{Eu}} (Y_b = 3, Y_b = 4) = \sqrt{(3 - 4)^2 + (9 - 6)^2}$$

$$5. D_{\text{Eu}} (Y_b = 4, Y_b = 5) = \sqrt{(4 - 5)^2 + (6 - 4)^2}$$

$$6. D_{\text{Eu}} (Y_b = 5, Y_b = 6) = \sqrt{(5 - 6)^2 + (4 - 8)^2}$$

Euclidean similarity coefficient = $1 - \sqrt{\{\text{Sum of all square of all six } D_{\text{Eu}} \text{ values using 1-NN}\} / \{\sqrt{6^2 + \text{Sum of square of all six } J_{\text{spi}} \text{ values}}\}}$

(vii) Euclidean similarity = $1 - \sqrt{2^2 + 17^2 + 2^2 + 10^2 + 5^2 + 17^2} / \sqrt{6^2 + 386} = -0.305$

Differing Similarity Coefficients for SPIs Calculated from Cosine distances and Euclidean Distances The following section explains the use of cosine distance and the situations in which D_{Cos} does not find similarity correctly.

Consider a comparison between the cosine and Euclidean similarities when finding similar items. Several situations exist in which predictions from two computational approaches differ. The reason is that triangular inequality holds true for Euclidean distances, while does not hold true for cosine distances.

Certain dimensions have widely different values. For example, let us compare sales JLRS and ZS in column 3 and 5 of Table 6.2. ZS values are nearly ten times the value of JLRS values. A solution is normalizing the values in all dimensions by dividing with the mean values using Equation (6.21). However, that also may give differing and incorrect results using D_{Cos} .

Cosine singularity is found to exhibit correct results for similarities in text documents. Cosine similarity is very efficient to evaluate situations of sparse vectors and those where one needs to consider non-zero values in the dimensions.

Concept of Sparse and Dense Vectors Sparse vector uses a hash-map and consists of non-zero values. Hash-map is a collection, which stores data in (key-value) format (Section 3.3.1). Format is also called random access. Hashing means to convert a large value or string into shorter value or string so that indexing for searching is fast.

For example, assume a vector, which consists of array elements, (subject,

number of students opting, average GPA).

1. Dense vectors have elements (Hive, 40, 8.0), (Java, 30, 8.5), (FORTRAN, 0, 0), (Pascal, 0, 0). Dense vector consists of all elements, whether the element value is 0 or not 0.
2. Sparse vectors will be two only with elements (4, 40, 8.0) and (3, 30, 8.5). Random access Sparse vector means access to elements (key, value pairs) using key. Sparse vector consists of elements for which key is such that value is not 0 (Section 3.3.1).
3. Sparse vector has an associated hash-map in form of a hash-table. First row— Pascal, 1, second row— FORTRAN, 2, third row— Java, 3 and fourth row— Hive.
4. Hashing is a process of assigning a small number or small-sized string indexing, searching and memory saving purposes. Hash process uses a hash function, which results into not-colliding values. In case of two colliding numbers, the process assigns a new number. Sequential access sparse vectors mean two parallel accessing vectors, i.e., one to access keys and the other for values.

6.4.4.5 Edit Distance

Edit distance D_{Ed} is a distance measure for dissimilarity between two set of strings or words. D_{Ed} equals the minimum number of inserts and deletes of characters needed to transform one set into another. Applications of edit distances are in text analytics and natural language processing, similarities in DNA sequences etc. DNA sequences are strings of characters.

Levenshtein suggested a method for finding edit distance, minimum number of operations of deletion, insertion or substitution of a character in a set of strings to transform one into another. The cost of substitution is taken as 2. Thus, edit distance from computation using that method is also called the Levenshtein method.³

6.4.4.6 Hamming Distance

If both U and V are vectors, Hamming distance D_{Ha} is equal to the number of

different elements between these two vectors. Recall Example 6.5 (iv) for Hamming distance between **Jspi** and **Zspi**. Hamming similarity-coefficient between car models Jaguar Land Rover and Zest is $(1 - 2/7) = 0.7$. [70%]

If **M** is a matrix, then D_{Ha} is equal to the number of different elements between the rows of **M** ignoring the columns.

D_{Ha} between two strings of equal length is the number of positions at which the corresponding characters differ. D_{Ha} is also equal to the minimum number of substitutions required to transform one string into the other. D_{Ha} is also equal to the minimum number of errors that need correction using transformation or substitution.

Hamming distance is therefore another distance measure for measuring the edit distance between two sets of strings, words or sequences.

Self-Assessment Exercise linked to LO 6.3

1. Why is triangular inequality in a distance measure important?
2. How will you compute Jaccard similarity coefficients between datasets for **Jspi** and **Zspi**. Use data Table 6.2 as the training dataset.
3. Why does similarity in documents computed?
4. Write applications of Euclidean, Jaccard, Cosine, Edit and Hamming distance measures.
5. Explain how Euclidean, Jaccard, Cosine and Hamming distance measures can be applied for analyzing the dataset given in Table 6.2?

6.5 | FREQUENT ITEMSETS AND ASSOCIATION RULE MINING

The following subsections describes frequent itemset mining, market basket model, association rules mining, and their applications.

6.5.1 Frequent Itemset Mining

Extracting knowledge from a dataset is the main goal of data analytics and data

mining. Data mining mainly deals with the type of patterns that can be mined. A method of mining is Frequent Patterns (FPs) mining method. Frequent patterns occur frequently in transactional data.

Frequent itemset refers to a set of items that frequently appear together, for example, Python and Big Data Analytics. Students of computer science frequently choose these subjects for in-depth studies. *Frequent itemset* refers to a frequent itemset, which is a subset of items that appears frequently in a dataset.

Frequent Itemset Mining (FIM) refers to a data mining method which helps in discovering the itemsets that appear frequently in a dataset. For example, finding a set of students who frequently show poor performance in semester examinations. *Frequent subsequence* is a sequence of patterns that occurs frequently. For example, purchasing a football follows purchasing of sports kit. *Frequent substructure* refers to different structural forms, such as graphs, trees or lattices, which may be combined with itemsets or subsequences.

FIM is one of the popular techniques to extract knowledge from data. The technique has been an essential part of data analysis and data mining. The extraction is based on frequently occurring events. An algorithm specifies a given minimum frequency threshold for considering an itemset as frequent. The extraction generally depends on the specified threshold.

FIM finds the regularities in data. Frequent itemset mining is the preceding step to the association rule learning algorithm. Most often the algorithm is used for analyzing a business. For example, customers of supermarkets, mail order companies and online shops use FIM to find a set of products that are frequently bought together. This provides the knowledge of important pairs of items that occur much more frequently than the items bought independently. A sales person can learn the pattern of what should be bought together for sales.

The analysis results in:

- Improvement of arrangement of products in shelves and on catalog pages
- Marketing and sales promotion
- Planning of products that a store should stock up

Frequent Item-sets Mining (FIM), applications of FIM, market basket model, mining the association rules, use of Apriori algorithm, evaluation of candidate rules, applications of association rules, and finding the association and similarity

- Support cross-selling (suggestion of other products) and product bundling.

6.5.2 Association Rule— Overview

An important method of data mining is association rule mining or association analysis. The method has been widely used in many application areas for discovering interesting relationships which are present in large datasets. The objective is to find uncovered relationships using some strong rules. The rules are termed as association rules for frequent itemsets. Mahout includes a ‘parallel frequent pattern growth’ algorithm. The method analyzes the items in a group and then identifies which items typically appear together (association) (Section 6.8). A formal statement of the association rule problem is:

Let $I = \{I_1, I_2, \dots, I_d\}$ be a set of d distinct attributes, also called literals. Let $T = \{t_1, t_2, \dots, t_n\}$ be set of n transactions and contain a set of items such that $T \subseteq I$. An association rule is an implication of the form, $X \rightarrow Y$, where X, Y belong to sets of items called itemsets ($X, Y \subset I$), and X and Y are disjoint itemsets ($X \cap Y = \emptyset$). Here, X is called antecedent, and Y consequent.

Explanation:

1. \subseteq means ‘subset of’, \subset means ‘proper (strict) subset of’, \cap means intersection and \emptyset means disjoint, no commonality in members.
2. Consider an If () then () form of a rule. The *If part* of the rule (A) is known as *antecedent* and the *THEN part* of the rule (B) is known as *consequent*. The condition is *antecedent*. Result is *consequent*.

6.5.3 Apriori Algorithm

Apriori algorithm is used for frequent itemset mining and association rule mining. Apriori algorithm is considered as one of the most well-known association rule algorithms. The algorithm simply follows a basis that any subset of a large itemset must be a large itemset. This basis can be formally given as the Apriori principle. The Apriori principle can reduce the number of itemsets needed to be examined. Apriori principle suggests if an itemset is frequent, then all of its subsets must also be frequent. For example, if itemset {A, B, C} is a frequent itemset, then all of its subsets {A}, {B}, {C}, {A, B}, {B, C} and {A, C} must be frequent. On the contrary, if an itemset is not frequent, then none of its

supersets can be frequent. This results into a smaller list of potential frequent itemsets as the mining progresses.

Support is an indication of how popular an itemset is. That is the frequency of the itemset for appearing in a database.

Assume X and Y are two itemsets. Apriori principle holds due to the following property of support measure:

$$\forall X, Y: (X \subseteq Y) \rightarrow s(X) \geq s(Y) \quad (6.24)$$

Explanation: \forall means for all, and \subseteq means ‘subset of’ and can be ‘equal to or included in’. Support of an itemset never exceeds the support of its subsets. This is known as the *anti-monotone property* of support.

The algorithm uses k-itemsets (An itemset which contains k items is known as a k-itemset) to explore $(k+1)$ -itemsets in order to mine frequent itemsets from transactional database for the Boolean association rules (If Then rule is a Boolean association rule, as it checks if true or false).

The frequent itemset algorithm uses candidate generation process. The groups of candidates are then tested against the dataset. Apriori uses breadth-first search method and a hash tree structure to count candidate itemsets. Also, it is assumed that items within an itemset are kept in lexicographic order. The algorithm identifies the frequent individual items in the database and extends them to larger and larger itemsets as long as those itemsets are found in the database. The frequent itemsets provide the general trends in the database as well.

6.5.4 Evaluation of Candidate Rules

Apriori algorithm evaluates candidates for association as follows:

C_k : Set of candidate-itemsets of size k

F_k : Set of frequent itemsets of size k

$F_1 = \{\text{large items}\}$

for ($k=1; F_k \neq 0; k++$) do {

$C_{k+1} = \text{New candidates generated from } F_k$

for each transaction t in the database do

Increment the count of all candidates in C_{k+1} that are contained in t

F_{k+1} = Candidates in C_{k+1} with minimum support

}

Steps of the algorithm can be stated in the following manner:

1. Candidate itemsets are generated using only large itemsets of the previous iteration. The transactions in the database are not considered while generating candidate itemsets.
2. The large itemset of the previous iteration is joined with itself to generate all itemsets having size higher by 1.
3. Each generated itemset that does not have a large subset is discarded. The remaining itemsets are candidate itemsets.

Figure 6.8 shows Apriori algorithm process for adopting the subset of frequent itemsets as a frequent itemset.

Apriori – Example

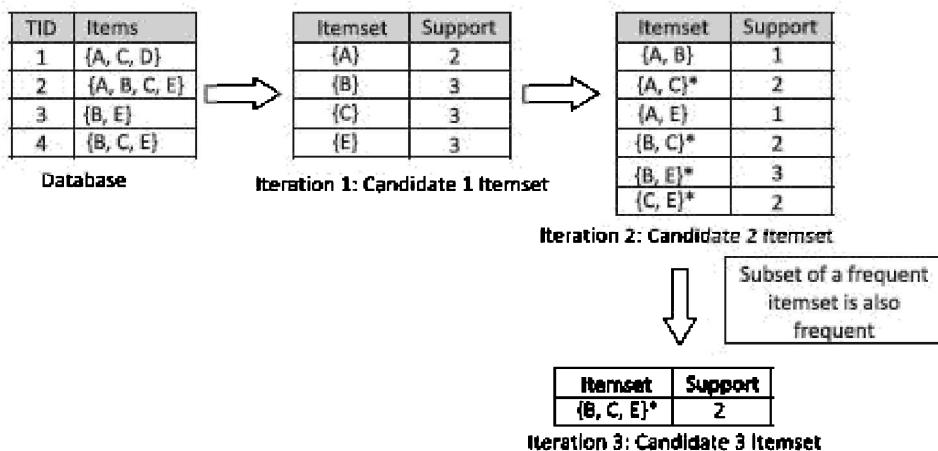


Figure 6.8 Apriori algorithm process for adopting the subset of frequent itemsets as a frequent itemset

It is observed in the Apriori example that every subset of a frequent itemset is also frequent. Thus, a candidate itemset in C_{k+1} can be pruned even if one of its subsets is not contained in F_k .

The Apriori algorithm adopts the fact that the subset of a frequent itemset is

also a frequent itemset. The algorithm thus reduces the number of candidates being considered by only considering the itemsets whose support count is greater than the minimum support count. All infrequent itemsets are pruned if they have an infrequent subset.

Apriori algorithm also possesses certain disadvantages. The algorithm requires multiple scans of a database. The process for generation of a complex candidate exploits more time, space and memory. Therefore, Big Data analytics need alternatives to Apriori algorithm to cut down on the size of candidate pairs. Section 7.4 will describe Park, Chen and Yu (PCY), multistage and multihash algorithms.

6.5.5 Applications of Association Rules

FIM is a popular technique for market basket analysis.

6.5.5.1 Market Basket Model

Market basket analysis is a tool for knowledge discovery about co-occurrence of items. A co-occurrence means two or more things occur together. It can also be defined as a data mining technique to derive the strength of association between pairs of product items. If people tend to buy two products (say A and B) together, then the buyer of product A is a potential customer for an advertisement of product B.

The concept is similar to the real market basket where we select an item (product) and put it in a basket (itemset). The basket symbolizes the transactions. The number of baskets is very high as compared to the items in a basket. A set of items that is present in many baskets is termed as a frequent itemset. Frequency is the proportion of baskets that contain the items of interest.

Market basket analysis can be applied to many areas. The following example explains the market basket model using application examples.

EXAMPLE 6.8

Suggest application examples of the market basket model.

SOLUTION

Application 1:

1. Items = Products

Baskets = Sets of products a customer purchases at one time from a store.

Example of an application: Given that, many people buy chocolates and flowers together:

- Run sales on flowers; raise price of chocolates.

The knowledge is useful when many buy chocolates and flowers together.

Application 2:

2. Items = Words

Baskets = Web pages

Unusual words appearing together in a large number of documents, for example, ‘research’ and ‘plastic’ may provide interesting information.

Market basket analysis generates If-Then scenario rules. For example, if X occurs then Y is likely to occur too. If item A is purchased, then item B is likely to be purchased too. The rules are derived from the experience. This may be the result of frequencies of co-occurrence of items in past transactions.

The rules can be used in several analytical strategies. The rules can be written in format If {A} Then {B}. The *If* part of the rule (A) is known as *antecedent* and the *THEN* part of the rule (B) is known as *consequent*. The condition is *antecedent* and the result is *consequent*.

If-then rules about the contents of baskets: $\{p_1, p_2, \dots, p_k\} \rightarrow q$ means, “If a basket contains all of p_1, p_2, \dots, p_k then it is likely to contain q .”

Scale of analysis:

- Amazon sells more than 12 million products and can store hundreds of millions of baskets.
- www has 1000 million words and several billion pages.

- 75 million credit card transactions in a month in India (RBI statistics of June, July 2016) at Point of Sales (POS) terminals.

Market basket analysis signifies shopping carts and supermarket shoppers at once. The analysis is the mining of transaction data to identify relations between different products. This is normally performed to identify products that a customer is likely to buy, given the products that they have already bought (or added to basket). The approach behind Amazon's users who bought a particular product also reviewed or bought other list of items is a well-known example of market basket analysis.

Applications of FIM in market analytics, medical analytics, web usage analytics, fraud detection, clickstream analytics

The applications of market basket analysis in various domains other than retail are:

- **Medical analytics:** Market basket analysis can be used for conditions and symptom analysis. This helps in identifying a profile of illness in a better way. The analysis is also useful in genome analysis, molecular fragment mining, drug design and studying the role of biomarkers in medicine. The analysis can also help to reveal biologically relevant associations between different genes. Further, it can also help to find the effect of environment on gene expressions.
- **Web usage analytics:** FIM approaches can be used with viewing data on websites. The information contained in association rules can be exploited to learn about website browsing of visitor's behavior, developing website structure by making it more effective for visitors, or improving web marketing promotions. The results of this type of analysis can be used to inform website design (how items are grouped together) and to power recommendation engines (Section 6.8). Results are helpful in targeted marketing. For example, advertising content that people are probably interested in, based on past behavior of users.
- **Fraud detection and technical dependence analysis:** Extract knowledge so that normal behavior patterns may be obtained in illegal transactions from a credit card database in order to detect and prevent fraud. Another example can be to find frequently occurring relationships or FIM rules

between the various parties involved in the handling of the financial claim. Some examples are:

- ◆ Financial institutions to analyze credit card purchases of customers to build profiles for fraud detection purposes and cross-selling opportunities.
- ◆ Insurance institution builds the profiles to detect insurance claim fraud. The profiles of claims help to determine if more than one claim belongs to a particular victim within a specified period of time.
- Click stream analysis or web link analysis: Click stream refers to a sequence of web pages viewed by a user. Analysis of clicks is the process of extracting knowledge from web logs. This helps to discover the unknown and potentially interesting patterns useful in the future. It facilitates an understanding of the behavior of website visitors. This knowledge can be used to enhance the way that web pages are interconnected or for increasing the sales of the commercial websites.
- Telecommunication services analysis: Market basket analysis can be used to determine the type of services being utilized and the packages customers are purchasing. This knowledge can be used to plan marketing strategies for customers who are interested in similar services. For example, telecommunication companies can offer TV Internet, and web-services by creating combined offers. The analysis might also be useful to determine capacity requirements.
- Plagiarism detection: It is the process of locating instances of similar content or idea within a work or a document. Plagiarism detection can find similarities among statements that may lead to similar paragraphs if all statements are similar and that possibly lead to similar documents. Formation of relevant word and sentence sequences for detection of plagiarism using association rule mining technique is also very popular technique.

6.5.5.2 Finding Association

Association rules intend to tell how items of a dataset are associated with each

other. The concept of association rules was introduced in 1993 for discovering relations between items in sales data of a large retailing company.

The following examples give rules between items found associated in the sales data of a retailer.

EXAMPLE 6.9

Suggest association rules between items found in the sales data of a retailer, and rules for course choice for a computer science student in college.

SOLUTION

1. $\{\text{Bread}\} \rightarrow \{\text{Butter}\}$

The rule suggests a relationship between the sales of bread and butter. A customer who buys bread also buys butter.

2. $\{\text{Chocolates}\} \rightarrow \{\text{a Gift Box}\}$

The rule suggests a that relationship between the sales of chocolates and empty gift boxes exists. A customer who buys chocolates also buys a gift box.

3. $\{\text{Java programming}\} \rightarrow \{\text{advanced web technology}\}$ and
 $\{\text{Python programming}\} \rightarrow \{\text{Big Data Analytics}\}$

The rules suggest relationships between Java and advanced web technology, and Python programming and data analytics. Students who opt for Java programming also want to learn advanced web technology, and those who opt for Python programming also opt for Big Data Analytics.

4. $\{\text{Data Mining}\} \rightarrow \{\text{Data Visualization}\}$

The rule may be that 90% of students who select data mining as a major subject will opt for the data visualization course as well.

5. $\{\text{Computer Graphics, Modeling Techniques}\} \rightarrow \{\text{Animation}\}$

The rule may be that students who study computer graphics and modeling techniques courses are likely to choose the course on animation in higher semesters.

Association analysis is applicable to several domains. Some of them are marketing, bioinformatics, web mining, scientific data analysis, and intrusion detection systems.

The applications might be to find: products that are often purchased together, types of DNA sensitive to a new drug, the possibility of classifying web documents automatically, geophysical trends or patterns in seismicity to predict earthquakes and automate the malicious detecting characteristics.

In medical diagnosis, for example, considering the co-morbid (co-occur) conditions can help in treating the patient in better way. This helps in improving patient care and medicine prescription.

6.5.5.3 Finding Similarity

Section 6.4 describes finding similarity of an item attribute, such as sales percentage increase using Euclidean or cosine similarity coefficients. Section 6.4.2 describes Jaccard similarity of sets. The similarity of sets applies to recommenders and collaborative filtering.

Let A and B be two itemsets. Jaccard similarity index of two itemsets is measured in terms of set theory using the following equation:

$$\text{Jaccard itemsets similarity index} = 1 - \frac{|A \cap B|}{|A \cup B|} \times 100\%. \quad (6.25)$$

Explanation: \cap means intersection, number of those elements or items which are the same in set A and B. \cup means union, number of elements or items present in union of A and B.

EXAMPLE 6.10

- (i) How will you define similarity in purchase of a car model?
- (ii) How will you specify frequent threshold for FIM? How will you use association rule to find and count the cities where more than threshold numbers buy a specific car model?

SOLUTION

- (i) Assume two sets of car customers, youth Y and family F. Assume in set

Y , 40 out of 100 youths and F 50 out of 200 families opted for the Tata Zest car model. Jaccard similarity index $J_{Zest}(Y, F) = 40 / (100 + 200).100\% = 13\%$. Two sets are sharing 13% of the members who purchased a Zest.

- (ii) FIM involves finding similarity index in large number of sets after specifying the similarity index threshold which defines an itemset as frequent. Assume N sets of car customers, youth Y_1, Y_2, \dots, Y_N , and N sets of families F_1, F_2, \dots, F_N in N cities. Assume that meaning of frequent is that 10% or more of $Y_i + F_i$ buying Zest among the various car models. Assume all other models sell less than that in the cities. Here $i = 1, 2, \dots, N$.

Let set X is a set, which has X_i as member if youth buy or if family buy the Zest car model frequently in the i^{th} City. Initialize value, $j = 0$ for frequent item sets. Then association rule for the FIM in the present case is:

If ($J_{Zest}(Y, i, F) > 10\%$) Then (City X_i is a member of X and $j = j + 1$) (6.26)

The rule is used for all cities for $i = 1, 2, \dots, N$; Here j is the number of cities where frequent item set {Youth, Zest}. FIM gives a set of j cities and youth, where youth buy Zest more than 10% of all car buyers.

Self-Assessment Exercise linked to LO 6.4

1. How does frequent itemset mining function mine the association rule?
2. Why does Apriori principle that 'if an itemset is frequent, then all of its subsets must also be frequent' hold true?
3. What are the features of Apriori principle that enable frequent itemsets mining?
4. How do you evaluate candidates for the associations?
5. How does concept of market basket model apply for frequent itemset mining?

6. List five examples where the association rule and count of frequent item sets apply.

6.6 | CLUSTERING ANALYSIS

The following subsections describe clustering and cluster analysis methods.

LO 6.5

Clustering a collection, cluster analysis, K-means and other methods, determining the number of clusters and cluster diagnostics

6.6.1 Overview of Clustering

Clustering of a collection means ‘a process (method) of grouping a collection of objects into subsets or clusters’ according to their distinct characteristics in the group. Clustering forms one or more clusters, such that objects within one cluster are similar to each other while the objects belonging to different clusters are dissimilar. The process can assign restriction on further additions of similar objects or add further new dissimilarity conditions. This limits the number of objects in a cluster in a collection.

Clustering and cluster analysis need segmentation of a population into a number of subgroups using unsupervised techniques of data mining.

For example, consider a university course. Assume a cluster of students that has distinct characteristic, i.e., students mostly get high grade points (GPs) in semester examinations. Input datasets consist of university course students (GPs) in semester examinations. Clustering algorithm computes the cluster from the input datasets only. The following example gives the results of cluster analysis for students with high GPs in both theory and practical courses.

EXAMPLE 6.11

Consider a superset D of all data of students enrolled S in a university. D consists of computer courses as members of set C. C consists of GPs in a semester examination as members of subset S. S consists of GPs as members of GPs in theory T subjects as well as practical P subjects.

Assume the following: Only one cluster with centroid exists at Theory GPA_T = 8.25 and Practical GPA_P = 8.25. The similarity criterion function is that GPAs in theory and practical both are high, i.e., (GPA_T, GPA_P) are within $= (8.25 \pm 1.75, 8.25 \pm 1.75)$. The number criterion function is when 8% and above students in the set C, then a cluster of high GPA_T with high GPA_P exists. This means the circle that surrounds the points within the cluster has periphery diameter = 3.5, twice of 1.75.

How does a plot show a cluster of members in S high GPAs in theory T as well as practical P subjects in number of university courses C?

SOLUTION

Consider plot of the P_GPAs and T_GPAs, GPAs in practical subjects as independent variable along the x axis and the GPAs in theory subjects as dependent variable along the y axis. Figure 6.9 shows cluster 1 and the results of cluster analysis of students with criterion 1. The cluster consists of students with high GPAs in theory T as well as practical P subjects in a number of university courses C in a semester S.

Each dot in Figure 6.9 corresponds to a distinct enrolled student in university computer courses (set C) for semester examination (set S).

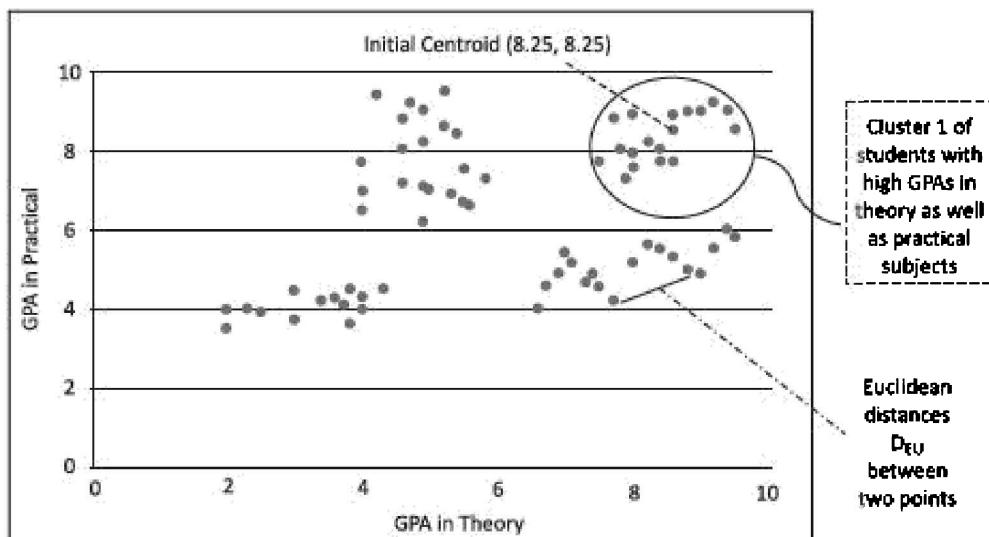


Figure 6.9 Result of cluster analysis of students' cluster 1 with high GPAs in theory subjects T as well as practical subjects P in

a number of university courses C and semesters

Partitions Example 6.11 considers one cluster. It considers one of the four possible partitions and assumes a starting centroid point= (8.25, 8.25). Opted criterion is considering distances up to 1.75 from a centroid for inclusion in the cluster of similar students. The figure also shows the centroid of the cluster at (8.25, 8.25) and a circle for criteria for the distances. Students in the cluster are those whose practical subject GPAs are between 6.5 and 10.0 and theory subject GPAs between 6.5 and 10.0.

Other three options can be as follows:

2— GPA in practical subjects between 4.5 and 6.0 and theory GPA between 6.5 and 10.0

3— GPA practical subjects between 6.5 and 10.0 and theory GPA between 4.5 and 6.0

4— GPA practical subjects below 4.5 and theory GPA below 4.5.

Centroid Figure 6.9 in Example 6.11 shows centroid (a central point of a cluster) GPAs of practical subjects (P_GPAs) and GPAs (T_GPAs) for data points of in subset S of set of students C. Example 6.11 considers centroid as point for the student sub-groups where means of both the GPAs (P_GPAs) and GPAs (T_GPAs) are high (near 8.0).

Distance Metrics A distance metric is Euclidean distance, D_{Eu} (Equations 6.20a and b). A circle of radius corresponds to maximum D_{Eu} around the centroid when using the criterion of inclusion in the cluster (Figure 6.9). When D_{Eu} is used, then the boundary data points lie on the circle.

A distance metric is Manhattan distance, D_{Ma} (Equation 6.20c). When D_{Ma} is used then the boundary points lie on a rectangle around the cluster.

Criterion Function Cluster 1 in the figure shows a circle, which specifies that all data points within the circle are at a distance 1.75 from the centroid 8.25. Criterion function for cluster 1 can also put addition criterion that more than 8% data points of all students in set C fall inside the cluster and have practical P and theory T GPAs values (data points) $(8.25 \pm 1.75, 8.25 \pm 1.75)$.

Input Vector Input column vectors of each data point in the figure have

elements in the metric space. The column vectors are **Y** (Year), **CC** (Course-code), **ID** (Student-ID), **SC** (Semester-code), **P_T** (subject type (practical or theory)), **SubjID** (Subject code) and **GP** (grade point).

Output Vector T_GPA (Grade point average of theory subjects), **P_GPA** (Grade point average of practical subjects) are output column vectors for input column vector **ID**.

Unsupervised Learning Clustering methods use unsupervised learning methods. Unsupervised learning refers to a process in which an ML algorithm does not use known outputs for the selected inputs for taking decisions or making predictions. A training dataset consists of outputs for selected inputs. This means that cluster computations *use input vectors only*.

Clustering Applications Clustering of a collection has applications in education, business analysis, sales analysis, customer groups analysis, resources planning, sports, astrology, fraud detection, production control and scientific investigation. Section 6.6.2.1 describes use cases.

Clustering a large dataset of performances of students has many applications. For example, student performance analysis which enables finding a subset of students with high performances practical and theory subjects and finding a subset of students as potential programmers from high performance in programming subjects.

Clustering Algorithms Following are the categories of clustering algorithms:

1. Partitions/centroid based K-means (Section 6.6.1), K-medoids, Fuzzy k-means, Mean-shift clustering and other related methods
2. Connectivity and spectrum based hierarchical clustering (Section 6.6.2). When closeness relates to connectivity then spectral clustering
3. Probabilistic distribution based Latent-Dirichlet-Allocation (LDA) (Section 6.9), Gaussian Mixture Model (GMM), Expectation Maximization (EM) clustering and others, [Expectation Maximization (EM) algorithm uses a set of parameters that maximize the probability of the chosen PDF for data as a metric.]
4. Dimensionality reduction based Principal Component Analysis (PCA) (Section 6.9)

5. Density based Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
6. Neural Networks/Deep Learning— Auto-encoders, self-organizing maps

The following Examples 6.12 and 6.13 explain the usages of clustering concept in two different cases.

EXAMPLE 6.12

How does clustering express the gene for a living cell that is undergoing a biological process?

SOLUTION

Genes are expressed differently whenever a living cell undergoes a biological process. Clustering of cells in the biological process enables the study of gene expression. This is required for understanding the underlying biological processes. This study is required to be carried out for different developmental phases, different body tissues, different clinical conditions and different organisms.

Recall Example 1.5. The following example explains how clustering analysis helps a ACVMs company.

EXAMPLE 6.13

- (i) Recapitulate Example 1.6(i) of an ACVM Company. The company sells chocolates of say, five flavours. How does the clustering concept help the company to plan future strategies?
- (ii) The ACVM company has to select new ACVMs in a city irrespective of whether these machines were installed or not previously. How does clustering guide the company?

SOLUTION

- (i) The company wants to analyze sales performance of all flavours of chocolates in order to check which flavour sells widely or which ACVM

needs filling frequently. The cluster analysis could further be extended for the sale of a particular flavour at many ACVMs in various cities. The clustering algorithm helps to find customer preferences in a specific set of regions.

The company needs to establish its sale points by putting its ACVMs in different regions. The location for installing ACVMs can be found using the clustering algorithm so that more of its customers receive a supply of their favourite flavours.

- (ii) The demand for new sales point is to be analyzed. Firstly, regions of city where youth population is high, such as regions with hostels and the regions of minimum concentration of other vendors need to be identified. Clusters of sparse or subserviced areas and clusters of higher sales potential need to be first identified for installing new ACVMs. Finding these options of course may require mathematical or statistical analysis.

The above examples require a study of problems based on grouping objects of similar types or characteristics. This requires applying exploratory data mining techniques for statistical data analysis. These techniques are used in many fields, including ML, pattern recognition, image analysis, information retrieval and bioinformatics.

Difference With Respect to Classification Clustering finds only the similar objects. Classification differs from clustering in the sense that classification assigns a class to each distinct set of characteristics in the collection. For example, classification will assign four classes of students, as per four criterion functions. For example, a collection of students in Figure 6.9 can be classified into one of the four classes: (i) good overall performing students, (ii) poor performance in practical subjects, (iii) poor performance in theory, and (iv) poor performance in both type of subjects. Section 6.7 describes classification and classifying methods in detail.

Clustering on the other hand discovers a large number of close-by points which form a distinct set in a collection. How much large and how much close depends on the chosen criterion function.

The following subsections describe selected clustering algorithms.

6.6.2 K-Means

MacQueen (1967) developed K-means algorithm. This is one of the simplest unsupervised learning algorithms for clustering. The algorithm groups the objects based on the attributes (features) into k number of groups where k is a positive integer number.

The grouping of data results into k clusters (C_1, C_2, \dots, C_k) represented by K centroids. A centroid is fundamentally a central representative of a cluster. The centroid of each cluster is the mean of all the instances belonging to that cluster. All objects of the cluster have similar characteristics, and fall within a criterion function specified for the cluster.

Criterion Function A criterion function assumes that at least p number of objects have similar characteristics, and are within the specified distances from a cluster centroid. A centroid may be assumed in the criterion, i.e., it is the one for which the sum of square of distances is least for all points of each cluster of a collection. The following example explains the use of K-means method.

EXAMPLE 6.14

How will you consider the criterion function in K-means method? Assume partitioning into k clusters. Assume the cluster problem similar to the one in Example 6.11.

SOLUTION

When partitioning N objects into k -clusters, the optimization of number of clusters and their centroid positions uses a criterion function. K-means method takes assumptions about similarity criterion. Assume that similarity is when T_GPAs are within 1.25 and P_GPAs also within 1.25. Assume that K-means method centroids, C_1, C_2, \dots, C_k are ones for which the sum of square of distances is least for all points taken into the clusters, C_1, C_2, \dots, C_k , respectively. This means that the sum is $1.25 \times 1.25 + 1.25 \times 1.25$ at the farthest point in an i^{th} cluster.

Assume criterion that $p = 0.08 \times N$, 8% of all students within a cluster.

K-means methods evaluates C_1 , C_2 , ... using the data points and criterion function assumptions. Considering Example 6.11, $C_1 = (8.25, 8.25)$. Another cluster C_2 may evaluate to $(8.25, 3.25)$. Number of objects in C_1 , C_2 ... may evaluate and come out to be 18%, 10%, but at least 8%.

6.6.2.1 K-means Use Cases

Clustering can also provide a significant way to solve a number of real-life situations. The following are the use cases where K-means is fast and efficient:

- Identifying abnormal data items in a very large dataset. For example, identifying potentially fraudulent credit card transactions, risky loan applications and medical claim fraud detection.
- The feature similarities information helps the K-means algorithm to be used in an image retrieval system
- Applied to many use cases in healthcare and helps to better characterize sub-populations and diseases by medical conditions. Some examples include:
 - ◆ Finding diabetic/non-diabetic or hypertension/non-hypertension group structure from the input value.
 - ◆ Identifying similar patients based on their attributes to explore costs, treatments or results.
 - ◆ To forecast the possible type (cause) of future treatment or hospitalization of affected patients.
 - ◆ Help researchers discover new insights by segmenting patients and providing them with effective treatments.
- To find the segment of customers and customer category using the spending behavior characteristic.

6.6.2.2 Overview of the K-means Method

Computations are needed for finding the distances between the data and the corresponding cluster centroid. A distance is taken as Euclidean, squared

Euclidean, Manhattan or Cosine distance (Equations (6.20a) to (6.20d) and Section 6.4.4).

Figure 6.10 shows the steps in K-means clustering.

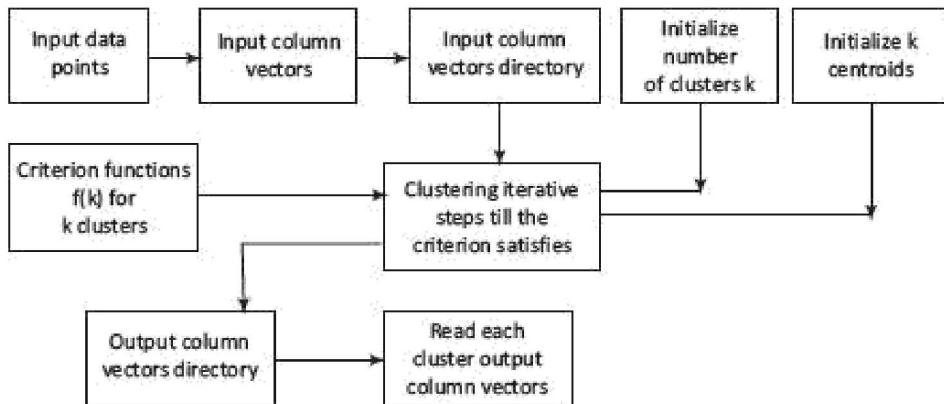


Figure 6.10 Steps in K-means clustering

K-means can be executed in the following steps:

1. Randomly initialize the k cluster centroid points ($= C_1, C_2, \dots, C_k$) as partition centers which mean partitions with these cluster centroids.
2. Go through each of the data points and assign points to a cluster where the distance from a centroid is minimum.
3. Identify the centroid of the new cluster formed. It is the average of all the data points in a cluster. In other words, the algorithm calculates the average of all the points in a cluster, and moves the centroid to that average location.
4. The process is repeated until no change in the clusters takes place (or possibly until some other stopping condition is met).
5. Steps of iterative relocation algorithm:
 - (1) Input: N (objects) and k (the number of clusters)
 - (2) Output: A set of k clusters, which uses criterion-functions $f(k)$ (For example, minimizing the sum of squared distances (Euclidean distances) for each cluster

(3) Algorithm steps:

- (i) Initialize k centroids as the initial solution.
- (ii) (Re) compute memberships for the objects using the current cluster centroids
- (iii) Update centroid of the cluster according to new memberships of the objects.
- (iv) Repeat from Step (ii) until there is no object change the cluster centroid.

Iterative methods compute the centroid values for each cluster. Centroids are the concentration points for clusters. The mean or median are typical choices for a centroid metrics.

Figure 6.11 shows the iterative method actions in K-means clustering (Consider students GPA example similar to Example 6.11).

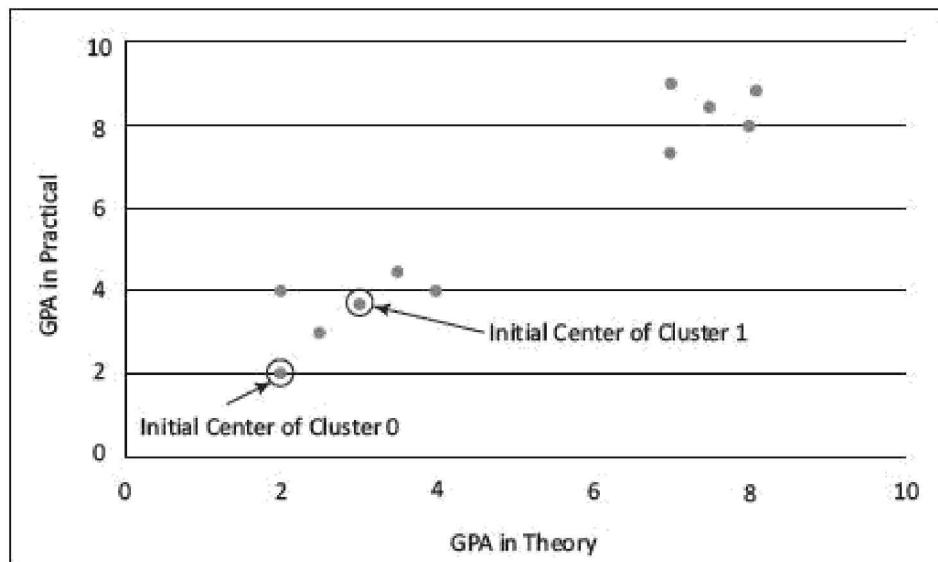


Figure 6.11 Iterative method actions in K-means clustering

Properties of K-means Clustering Algorithm The properties of K-means clustering algorithm are:

1. Number of clusters which form are always k clusters, where k is the number of partition centers.

2. Each cluster consists of at least one object in each cluster.
3. The clusters are flat (non-hierarchical) and they do not overlap.
4. Every member object of a cluster is closer to its cluster than any other cluster.

The algorithm has a number of variations, depending on the method for selecting the initial centroids, the choice for the measure of similarity, and the way that the centroid is computed. Most commonly, the Euclidean data exploits the mean as the centroid and selects the initial centroids randomly. The four other features of K-means clustering algorithm are as follows:

1. The K-means method is numerical, unsupervised, non-deterministic and iterative
2. The method is well suited if the clusters are globular
3. The centroid depends on the distance function that is measured by Euclidean distance (Sum of Squared distance (SSD)), cosine similarity or correlation
4. Centroid is the mean of the points in the cluster for SSD and cosine similarity; the median for Manhattan distance.

The K-means algorithm converges to a solution, which is typically a local minimum. The *space requirements* of the algorithm are $O(n \times d)$, where n is the number of points and d is the number of attributes. Here, only the data points are stored. The *time requirements* are $O(n \times k \times I \times d)$, where k is the number of clusters, and I is the number of iterations required for convergence. I is usually small as most of the convergence happens in the first few iterations.

Thus, the time required by K-means is efficient, as well as simple, as long as the number of clusters is significantly less than n .

$O(f(n))$ measures the efficiency of an algorithm in terms of function $f(n)$. If $f(n) = n^2$, then the requirement of the algorithm is proportional to n^2 . The requirement may be measured for memory or may be for run time. Space requirement $O(n \times d)$ means that memory taken by the algorithm is proportional to $n \times d$. O is called as the big O notation.

Advantage of K-means is that computing the distances between points and

group centres has linear complexity $O(n)$. Disadvantages are (i) need to choose k , the number of groups/classes required to form the clusters, (ii) need to start and randomly choose the cluster centres, the results may be choice dependent. Thus, there is less consistency of the results compared to other methods.

6.6.2.3 K-medoids Algorithm

A medoid is similar to a mean or centroid, but restricts to members of the dataset. A dataset may have more than one medoid. The K-medoids algorithm initializes k data points as exemplars (centers), which shift iteratively for minimizing dissimilarities. The algorithm K-medoids does clustering using an algorithm, which has flavours of k-means algorithm and medoid-shift algorithm.

1. Step 1: Choose a set of medoids.
2. Step 2: Compute distances from each medoid to other points.
3. Step 3: Cluster the data points according to their similarities with the medoid.
4. Step 4: Optimize the set of medoids using iterative process.

The sum of pair of dissimilarities minimizes in K-medoids compared to minimizing the sum of squared Euclidean distances. The algorithm is based on partitioning technique of clustering, which clusters the data set of n objects into k clusters.

Use of medoid is in graphs and other non-metric spaces. Non-metric means non-quantifiable. Medoids mean the objects in a cluster or dataset such that average of dissimilarities minimizes taking all cluster members (objects). Recall that distance is a measure of dissimilarity. Computation of minimum dissimilarity considers minimum distances of all pairs of points within a cluster. Median is v -dimensional data point.

6.6.2.4 Determining the Number of Clusters

K-means algorithm finds k clusters in a given dataset. The K-means algorithm partitions the objects into k non-empty subsets. Thus, k signifies assumption about formation of a number of clusters. The data points represent the objects. Choice of k is either random or as per specific initial starting data points that the user specified.

A partition technique generates specific number of flat disjoint clusters (say, k clusters). Each object belongs to a specific cluster. Each object in a cluster is closer to the centroid than to the centroid of any other cluster. The centroid can be an arithmetic mean of the attribute values of all the objects in case of real-valued data. The centroid can be the rank value of the objects in case of categorical data. The iterative relocation algorithm is an excellent method for finding k partitions/clusters/centroids.

When partitioning n objects into k clusters, optimization uses a criterion function. For example, criterion that more than 10% of all students within (8.25 ± 1.25 , 8.25 ± 1.25) GPAs in practical and theory belong to the same cluster (Example 6.11).

A useful tool for determining k is the silhouette value, s. A silhouette value computes from the similarity of an object with own cluster objects (means cohesion in the objects) instead of other clusters (means separation in the objects of all other clusters).

The s ranges from -1 to +1. Value -1 represents complete separation and +1 means complete cohesion. +1 means an object matches perfectly with an object in its own cluster and completely mismatches outside the cluster objects. When most objects of a cluster have high silhouette value, it means the cluster is well configured.

6.6.2.5 Diagnostics Method

Clustering validation tool does cross validation and validates initial centroid choices. A diagnostic tool uses the output of clustering for the decisions. For example, Microsoft Windows Server 2003 resource kit includes ClusDiag.exe, a tool for cluster diagnostics and verification tool for web files and events. It does basic verification and analyses configuration. It collects logs of files and events.

6.6.2.6 Reasons to Choose and Cautions

K-means algorithm finds k clusters in a given dataset, but one of the important questions is about the value of k to be selected. It is suggested to choose the value of k randomly or define it based on a domain requirement. However, if k + 1 clusters do not make significant change in the data points of clusters, from the case with k clusters do not increase one more cluster. Thus, select an optimal value of k such that introducing a new cluster may just be of little benefit.

Clustering explores similarities or cohesiveness in a significant number of objects or elements of the vectors. A point to remember is that improper criterion function can lead to erroneous results for clusters. One should be cautious while making predictions using current data points. For example, consider prediction from clustering analysis in Example 6.11. Clustering is not including teachers and resources availability for studying computer courses. Non-inclusion can lead to erroneous predictions.

6.6.3 Hierarchical Clustering

'Hierarchical clustering algorithms create a hierarchical decomposition of objects of a given data set using some criterion'. Figure 6.12 shows the original object points and one hierarchical cluster representation of those object points.

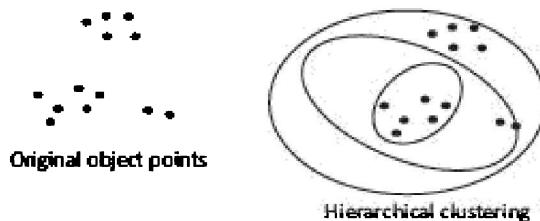


Figure 6.12 Original object points and one hierarchical cluster representation of those object points

EXAMPLE 6.15

How will you consider hierarchical clustering to solve the problem of ACVM owner for identifying distinct groups in their customer bases?

SOLUTION

Suppose the ACVM owner want to identify distinct groups in their customer bases, and then use this knowledge to develop targeted sales and marketing programs for a particular flavour of chocolate. They definitely require the formation of a hierarchy of nearest ACVMs targeting high sale campaigns for that flavour of chocolate. The hierarchical clustering may solve their problem efficiently. Recall Example 1.6. Figure 6.13 shows hierarchical clustering. The hierarchy is (i) clusters of C city-regions showing high sales per day and (ii) clusters of j set of regions R₁ and R₂ showing high total

sales per day.



Figure 6.13 Hierarchical clustering of (i) original object points in city C showing high total sales per day, (ii) clusters of j set of regions R1 and R2 showing high total sales per day.

Hierarchical clusters may correspond to meaningful taxonomies as well. For example, evolutionary relationships among animals in biological sciences, and product catalogs in the web (online) world. It can also be used to represent file systems of any operating system.

Hierarchical clustering generally produces consistent results. The results of hierarchical algorithms are represented by a tree-structured graph known as a dendrogram. The dendrogram basically records the sequences of merges or splits.

The individual objects are arranged along the bottom of the dendrogram. They are referred to as leaf nodes. Object clusters are formed by joining individual objects or existing object clusters with the join point referred to as a node. Figure 6.14 shows dendrogram structure nodes. Each dendrogram node has a right and left sub-branch of clustered objects. The vertical axis is labeled as distance, and is used for distance measures between objects or object clusters. The height of the node represents the distance value between the right and left sub-branch clusters.

- Each node represents a group.
- Root node represents the group containing complete data set.
- Leaf node represents single object of the data set.
- Internal node has two children (sub-branches) representing the internal groups or leaf nodes (objects).

As the name implies, hierarchical clustering builds a hierarchy of clusters. Any number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level.

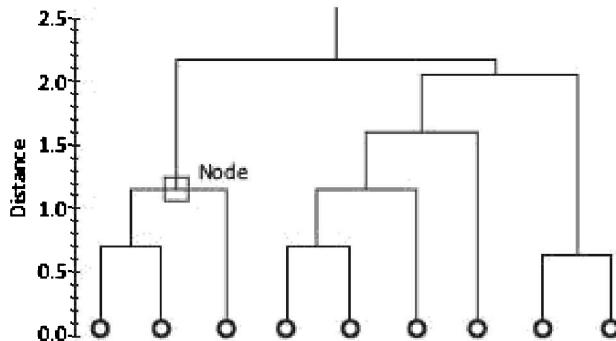


Figure 6.14 Dendrogram structure

There are two ways to do this: one is to start from the bottom, with all the objects as clusters and then, at each step, merge the two of them. This is known as agglomerative (bottom-up) hierarchical clustering. The other one is called divisive (top-down) hierarchical clustering and starts from the top, with all the objects in a big cluster, and at each step performs a split. Both the methods produce dendograms.

Agglomerative Clustering Algorithm is the more popular hierarchical clustering technique. The computation of similarity is the key operation. The basic algorithm is straightforward.

1. Consider each data object as a cluster.
2. Compute the similarity between each pair of objects (cluster).
3. Repeat until only a single cluster remains:
 - (i) Merge the two clusters having the smallest dissimilarity
 - (ii) Update the similarities between a pair of clusters.

Online Contents (OLC) for BDACh06OLC6_1 for hierarchical clustering, which accompany this book describe in detail different approaches including agglomerative-divisive-distance measures for defining the distance between clusters in the different algorithms (Solution of Practice Exercise 6.12).

A dataset consisting of n object points the space and the time requirement for

agglomerative hierarchical clustering is:

1. $O(N^2)$ space to store the distance matrix, most of them are $O(N^2)$ or more but one can stop at any arbitrary number of clusters.
2. $O(n^3)$ time in most of the cases. There are n steps and at each step the size n^2 distance matrix must be updated and searched. The complexity can be reduced to $O(n^2 \log(n))$ time for some of the approaches by using appropriate data structures.

Advantage of hierarchical clustering is in finding the underlying finer structure. For example, resource planning for filling the ACVMs or for installing ACVMs in appropriate regions of the city (Figure 6.13).

Hierarchical clustering is not used in Big Data environment because of scalability and partition ability issues. Another disadvantage of hierarchical clustering in Big Data analytics is a large n and the time complexity of $O(n^3)$ compared to $O(n)$ linear complexity in K-means and GMM.

Self-Assessment Exercise linked to LO 6.5

1. Write meanings of partition, input vector, output vector, centroid and minimizing distance method.
2. Why is clustering an unsupervised machine learning method?
3. What does a machine learn during running of a clustering algorithm?
4. How do the cluster size and criterion function relate?
5. How is minimization performed using Euclidean squared distance for finding the cluster centroids?
6. Write effects of using in Euclidean squared, Euclidean distance and Manhattan distance used in K-means clustering on cluster size and centroid. How does it effect the cluster size?
7. Why does time complexity of K-means equal $O(n)$?
8. When is hierachial clustering useful?

6.7 CLASSIFICATION

Classification refers to learning from existing categorizations and forming groups of objects showing similar characteristics. For example, categorize students good in theory and practical subjects both as 'very good'.

Classification is a supervised learning method. Classifier is an ML algorithm for classification, which decides usage of the experience, and emulates certain human decisions.

The classification techniques are used in many fields, including machine learning, pattern recognition, image analysis, information retrieval and bioinformatics. Classification is an exploratory data-mining method, which creates groups of objects of similar types or characteristics.

Consider an automatic chocolate vending machine (ACVM) for vending chocolates of say, five flavours.

The chocolate company needs to establish its sale points by putting its ACVMs in a particular region. The location of putting these ACVMs can be found by a clustering algorithm so that all its ACVMs receive supply based on the analysis of customers favourite flavours.

The company surveys the sales performance of each flavour in order to check which flavour is giving wider sales, and needs enhanced supply frequency to ACVMs. The survey could be further extended for the sales promotion of a particular flavour in its various cities.

6.7.1 Concept of Classification

A classifier needs training, which means learning from existing categorizations and forming groups of objects showing similar characteristics. Training is a learning process which uses training dataset T and generates a model program, M. *Training dataset* means a subset E of an exemplary dataset which includes training variables. T includes value of the target variables and predictors also. The training algorithm generates a 'Model', which is a program which gives the output vectors for taking the decision of the class to which the input vector belongs. (Remember, a set of data points can be represented by a vector in v-

LO 6.6

K-nearest neighbour (KNN) classifier, decision trees, Random-Forest classifier, AdaBoost and other ensemble classifiers, Naïve Bayes classifier and SVM-based classifier

dimensional space.)

Figure 6.15 shows the steps during the learning phase of a classifier. The learning needs (i) training dataset T, which includes P, both as the inputs. The classifier consists of a training algorithm and creates a model program from inputs and outputs ET (estimated target variables). The algorithm creates a model program M for internal uses of a classifier and its copy M'. The M' estimates target variable(s) are inputs to a decider D program to decide which data points to put in which class. It emulates certain human decisions for classification.

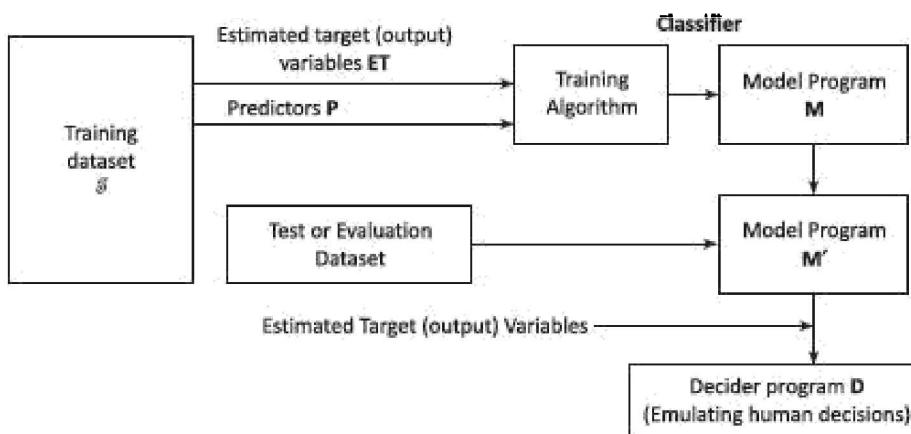


Figure 6.15 Steps during the learning phase of a classifier

A dataset for testing or evaluating tests the decisions of the Model and Decider. The test dataset is a subset, whose members are the input vectors, Predictors P and target output vectors (variables) ET for taking the decision for the class to which input vectors belong. If decision from model passes the test, then the model will predict correct decisions and the class from future inputs.

Predictor can be: (i) a continuous value, such as grade point; (ii) text, such as Java; (iii) string, such as 'GPA ≥ 8.0'; and (iv) a category, such as 'very good', 'potential researcher', 'high performance', 'Zest model', 'red apple'.

The following example gives the understanding of steps in classification of students with high GPs in both theory and practical courses. Consider a classifier for classifying 'very good performing students' and thus, who are potential innovators.

EXAMPLE 6.16

Recall Example 6.11. Consider the student ID, his/her T_GPAs in theory subjects and P_GPAs in practical subjects. GPAs means grade points average computed from grade points in the semester examination. Assume the data points similar to ones in Figure 6.9.

How will you specify: (i) training dataset, (ii) target variables, (iii) predictors (predictor variables), (iv) features, (v) training algorithm, (vi) model program, (vii) exemplary (Test) dataset, (viii) estimated target variables, and (ix) decider which categorizes and forms the classes (groups of objects) in a classifier? (x) What do you mean by field in a record of data points?

SOLUTION

- (i) Training dataset: 75% of the student data consisting of student ID, his/her T_GPAs in theory subjects and P_GPAs in practical subjects.
- (ii) Target means, a feature as a target variable which is found using learning examples. Target is to find whether a student group belongs to one with high GPA in theory and practical subjects, and both. Target variables are T_GPAs and P_GPAs.
- (iii) Predictors (Predictor variables): Subject of study, subject-wise attendance in class, number of hours a student studies a subject etc.
- (iv) Features: Gender, Age, Coaching (Yes/No), Residence (Rural/Urban), High T_GPAs and P_GPAs
- (v) Training algorithm steps are as follows:
 - Define a training set.
 - Choose a learning algorithm. For example, Support Vector Machines, Naïve Bayes or Decision Trees.
 - Complete the design. Run the algorithm.
 - Evaluate the accuracy of the learned model from test dataset.
- (vi) Model Program: Classification technique (or classifier) which systematically builds the classification model from an input dataset.

- (vii) Test dataset: Remaining 25% of the student data consisting of student ID, his/her T_GPAs in theory subjects and P_GPAs in practical subjects.
- (viii) Estimated target variables: GPA in semester examinations.
- (ix) Record of data points: Record is a container for T which consists of fields. For example, training grade sheets of the students is a record.
- (x) Fields in the record: Fields are a part of a record. A field consists of a value, feature, characteristic, outcome or category. Example of input fields are T_GPA, P_GPA, Semester, High T_GPAs and P_GPAs in input vectors at a student record. Example of feature variable fields are 'Excellent', 'Potential Programmer'.

6.7.1.1 Concept of Supervised Learning

Methods of ML are fundamentally driven by data. On the basis of data availability these methods are broadly classified into supervised learning and unsupervised learning. *Supervised learning* refers to a case when an algorithm uses training data to take decisions or make predictions.

Supervised learning uses a known output dataset for the input dataset (called the training dataset). The Model (program) then learns to make predictions. The output datasets are used to train the machine and get the desired outputs. Test dataset tests the Model and Decider to verify themselves. The developed Model makes predictions for an unknown output for the further input datasets. However, in unsupervised learning, no output and input datasets are provided to train the machine.

Table 6.3 illustrates the difference between supervised and unsupervised learning.

Table 6.3 Supervised vs unsupervised learning

	Supervised Learning	Unsupervised Learning
Training set	Used	Not used
Input	Observations	Latent variables

variables		
Output variables	Observations	Observations
Labels	All data are labeled	All data are unlabeled
Goal	To approximate the mapping function exactly in a way that on new data input, it can predict the output variables for that data.	To model the underlying structure or distribution in the data in order to learn more about the data
Application examples	Classification and regression problems	Clustering and association problems

The following example shows how classification forms the groups of objects with similar characteristics from analysis of student performances in different courses.

EXAMPLE 6.17

Show the results of a classifier for classification of students showing potential as scientists and researchers, data architects and programmers after analysis of the GPAs in different courses, i.e., theory and practical.

SOLUTION

Figure 6.16 shows results of classification of student groups into courses as potential classification into classes (groups) of students in different courses as potential (1) scientists and researchers, (2) data architects and (3) programmers.

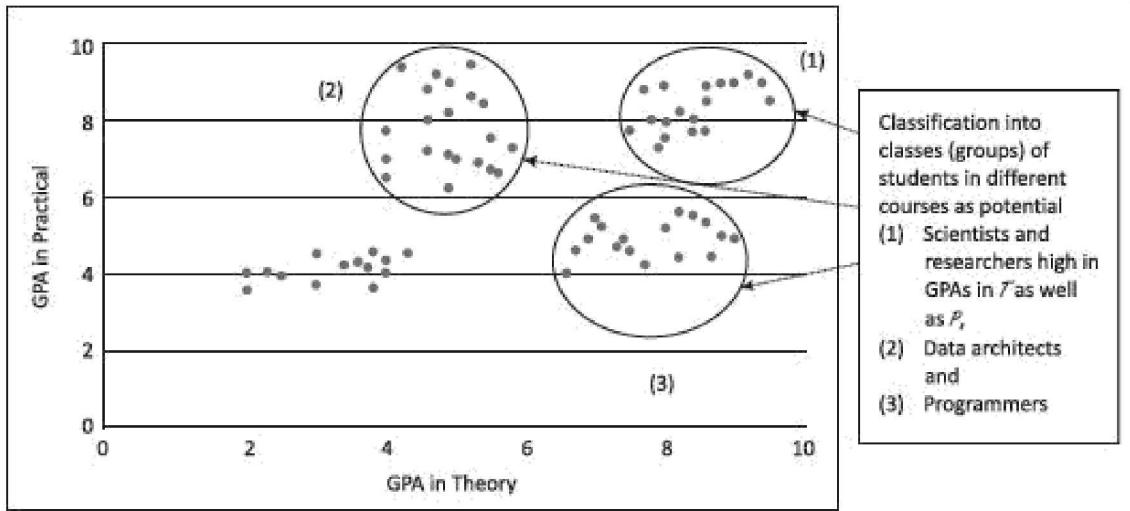


Figure 6.16 Classification on the basis of performances of the student groups

6.7.1.2 Clustering and Classification Differences

Clustering and classification differ as shown below, though both the methods characterize the objects into groups by one or more features. Classification is a supervised learning method, whereas clustering is an unsupervised learning method used to form groups of objects with similar characteristics.

Table 6.4 compares and highlights the characteristics of classification and clustering.

Table 6.4 Classification vs clustering

Property	Classification	Clustering
Supervision	Supervised learning	Unsupervised learning
Training set	Used	Not used
Labels	All data are labeled	All data are unlabeled
Datasets	Consist of attributes and class labels	Consists of attributes
Process	Employs algorithms to categorize new data according to the observations of	Statistical concepts are used. Datasets are split into sub-sets

	the training set	with similar features
Goal	To find which class a new object belongs to from a set of predefined classes	To group a set of objects in order to find the relationship between them

The category structure is known in classification task, whereas the clustering deals with object collections, whose class labels are unknown.

Examples of classification techniques include decision tree classifiers, rule-based classifiers, Support Vector Machines (SVM) and Naïve Bayes Classifier (Section 6.7.4). Each classifier employs a learning algorithm to build a model that best fits the relationship between the objects of a class.

6.7.1.3 Classifiers

Classifiers for Big Data analytics require parallel and scalable computations with shared-nothing architecture for efficiency. Parallel means computations of the same set of codes simultaneously on multiple data nodes (Section 1.2.1). Scalable means linear relation between data volume and the required total number of computation steps and thus, computational time. This means if 10 MB data require time = T for processing, then 100 MB will take 10T. Shared nothing means during computations, no inter-processor communications, thus processors spend no time on them (Example 2.1).

Naïve Bayes and complementary Naïve Bayes are efficient between medium to large datasets (>1 M up to 100 M), but mostly suitable for text data variables and medium to high overhead for training. **Random Forest** uses all four types of predictor variables: continuous, text, words or categorical. Random Forest is able to handle conditional and non-linear relationships and thus, the complex classification problems. It exhibits high performance computations *above 10 M dataset*.

Certain classifiers do sequential computations with shared architecture. For example, **Support Vector Machines** (SVMs). The SVM classifier is efficient for computational needs of small (0.1 M or less) to medium (< 1 M) datasets.

Stochastic Gradient Descent (SGD) algorithms, such as logistic regression are sequential, incremental efficient (fast) and used when computational needs are of small (< 0.1 M) to medium (< 10 M) dataset. Predictor variables can be of any of the four types. Section 6.7.3 describes SGD.

Hidden Markov and **multi-level perceptron** are also sequential algorithms. OLCs accompanying the book describe the *Hidden Markov* and *multi-level perceptron* (Solution of Practice Exercises 6.15 and 6.16).

6.7.2 K-Nearest Neighbour Classifier

Recall Sections 6.3.6 and 6.4.1.1 for applications of K-NN in regression and in similar items search (k is 1 for nearest neighbour, 2 for next to nearest and 3 for next to next nearest). Training dataset consists of k-closest examples in feature space. Feature space means, space with categorization variables (non-metric variable). For example, Grade A, B, C, D awarded in an examination are feature space variables. The k-NN learning is learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset. An object classification criteria is majority vote. k-NN is also called the lazy algorithm. The following example explains this method.

EXAMPLE 6.18

Assume that when CGPA (cummulative grade point average) is 8.0 and above up to the maximum 10.0), a student performance is classified as A. When CGPA is 6.5 and above below 8.0, performance is B. When CGPA is 5.0 and above below 6.5, performance is C. When CGPA is 4.0 and above below 5.0, performance is D. When CGPA is less than 4.0, performance is F (poor).

A training dataset consists of data of 40 students. Training dataset consisting of vectors 1, 2, ... to 40 are (StudentID1, 8.1, A), (StudentID2, 7.6, A), (StudentID3, 6.6, B), (StudentID3, 4.6, D), (StudentID4, 8.8, A), (StudentID2, 5.6, C), (StudentID5, 6.7, B), (StudentID6, 5.2, C), (StudentID7, 4.5, D), (StudentID8, 7.9, B), (StudentID9, 6.8, B), (StudentID10, 9.1, A), (StudentID11, 7.1, B), (StudentID12, 7.9, B), (StudentID13, 6.0, B), (StudentID14, 8.0, A), ...

How will a 1-NN classifier classify a student of ID 250 with CGPA 7.2?

SOLUTION

The training algorithm will follow the following steps (Fig. 6.15):

- (i) Find the distance for each value of CGPA, distances of CGPAs and the

output variable class, A, B, C, D, or F from the training dataset of 40 students.

- (ii) Assign frequencies for class = A, B, C, D, or F for each range of grade points.
- (iii) Build and update the table with the additional datasets given of training, if required.

Table 6.5 gives guidelines of the table between the range of CGPA and frequency of output variables = A, B, C, D and F.

Table 6.5 Range of CGPAs and frequency of output variables = A, B, C, D and F.

CGPA Minimum	CGPA Maximum	Freq (A)	Freq (B)	Freq (C)	Freq (D)	Freq (F)
8.a	9.b or 10.0	> 0	0	0	0	0
6.c	7.d	0	> 0	0	0	0
5.e	6.f	0	0	> 0	0	0
4.g	4.h	0	0	0	> 0	0
0.i	3.j	0	0	0	0	> 0

The values of a, b, c, d, e, f, g, h, i and j build up gradually as the number of training input vectors and output vectors keep increasing, eventually 8.a reaching 8.0, 9.b reaching > 9.9 and so on. Time taken and the number of inputs and output vectors may be such that final output values may take a long time.

Model develops a program to read the table and compute output variable estimate for test dataset and unclassified input vector.

Decider Program: When input vector is for ID250, find the distance of 7.2 with the input vectors.ID1 distance is 0.9. Successive input vector distances from vector I1 to I40 are 0.9, 0.4, 0.6, 2.6,0.8, 1.6, 0.5, 2.0, 2.7, 0.5, 0.4, 1.9,0.1, ..., Nearest neighbours have distances = 0.4 from studentID 250.Using CGPA range and student CGPA, the decider will output the predictor variable as B.

Classifier places that student in class B.

6.7.3 Stochastic Gradient Descent Method – Logistic Regression

Stochastic in English means a process or system connected with random probability, chance or randomness. *Gradient* equals to change in a function value with respect to a very small change in a parameter value. For example, in a function $y(x_1, x_2, \dots, x_v)$ in v -dimensional space, gradient of y with respect to x_i equals differentiation of y with respect to $x_i = dy/dx_i$, where $i = 1, 2, \dots, v$.

Gradient descent means decremental change in gradient. Gradient descent is used for reaching convergence for each value of x_i iteratively. It reaches a minimum value for each x during optimizing the set of parameters or values which are input to an objective or other function. The gradient descends iteratively to a minimum value. Incremental or decremental change means successive increment or decrement in x that leads to approach towards the optimum value of y .

Recall Section 6.3.3. It explained how the best fit could be reached by using the ‘least squares criterion’, which says that best fit is one, which ‘minimizes the sum of the squared prediction errors.’ Here, Object function $Q = \sum_{i=1}^n (y_i - \hat{y}_i)^2$. Equation (6.15) is minimized to obtain coefficients of the regression equation (6.14), $\hat{y}_i = b_0 + b_1 x_i$ which gives best fit, i.e., minimizes Q . To minimize $Q = \sum_{i=1}^n (y_i - (b_0 + b_1 x_i))^2$ take the gradient (derivative) with respect to b_0 and b_1 , set to 0, respectively, and get the ‘least squares estimates’ for b_0 and b_1 from Equations (6.16) and (6.17).

The method is called logistic regression when we consider a generalized objective function as follows:

$$Q(\mathbf{y}) = N^{-1} \left[\sum_{i=1}^N Q_i(\mathbf{y}) \right]. \quad (6.27)$$

where N is the number of data points summed using the input vector, Q_i is i -th observation of y for input variables \mathbf{y} , being estimated such that the sum is minimized and parameters r coefficients are optimized.

Steps in an SGD algorithm are: (i) choose a starting input vector \mathbf{Y} and a learning rate e , i.e., the rate by which y decrements in next computations to get the minimum sum (Equation 6.27) and minimized regression coefficients for

computing Y , and (ii) randomly change (scuffle) the exemplary input vectors and corresponding output vectors which are used as examples for learning. For i -th observation 1 to m , find

$$y = y - b_1 \nabla Q_i(y) \quad (6.28)$$

Here, ∇ is the mathematical symbol for gradient, which computes gradient of $Q_i(y)$. b_1 is the linear regression coefficient in Equation (6.28).

SGD classifier trains and learns the computation of objective function values and classifies based on predictor values for each class.

Logistic regression uses hash values for the features, which means training algorithm assigns each feature a hash value, which is used for indexing, search and predictor variable.

OLCs accompanying the book gives examples of the Stochastic gradient descent method for logistic regression and classification (Solution of Practice Exercise 6.13).

6.7.4 Decision Tree Algorithm

Tree-based learning algorithms are simpler and efficient supervised learning methods. They provide accurate, persistent and ease of analysis to predictive models. Tree-based algorithms are used for solving classification and regression problems. They are suitable for representing non-linear relationships as well. Some of examples of tree-based learning algorithms are decision trees, Random Forest and gradient boosting.

Following are the important terms related to Decision Trees:

1. *Root Node*: Represents the entire dataset.
2. *Splitting*: A process of dividing a node into two or more sub-nodes
3. *Decision Node*: When a sub-node splits into further sub-nodes
4. *Leaf/Terminal Node*: Nodes that do not split further
5. *Pruning*: Process of removing sub-nodes of a decision node (opposite of splitting)
6. *Branch/Sub-Tree*: A sub-section of the entire tree

7. *Parent Node*: A node divided into sub-nodes
8. *Child Node*: A node derived from a parent node.

Decision Tree is a supervised learning algorithm, having a desired response value that is mostly used in classification problems. Decision tree works for both categorical and continuous input and output variables.

The decision steps are as follows:

First, split the dataset when classifying a response variable. That is based on most significant splitter/differentiator in input variables into two or more subsets.

The decision trees segregate the datasets based on all values of three variables and identify the variable, which creates the best homogeneous sets of datasets (which are heterogeneous to each other). The following example explains how decision trees segregate student datasets based on all values of three variables.

EXAMPLE 6.19

Consider a dataset of 100 students with three variables Gender (Boy/Girl), Branch (CS/EC) and GPA score of previous year (8.0 to 10.0). 40 out of these 100 enrolled in coaching class for learning the programming. It is required to create a model to predict who will enroll in coaching class. (CS: Computer Science, EC means Electronics and Communication).

SOLUTION

First segregate the students who took admission in coaching classes based on each significant input variable and find which variable is most significant among the three (Gender, branch, GPA).

Figure 6.17 suggests that variable GPA identifies the best homogeneous sets compared to the other two variables.

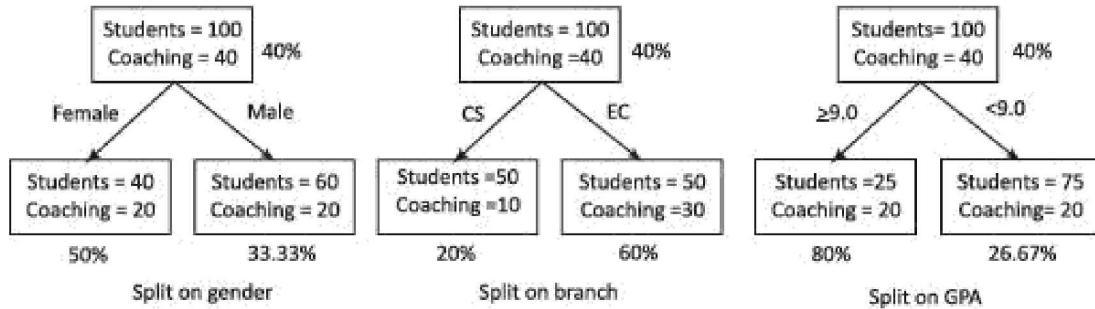


Figure 6.17 Example of decision trees for creating the best homogeneous sets of students

Types of decision tree are based on the type of response variables:

1. **Categorical-variable Decision Tree:** Decision tree, which has categorical response variable. For example, consider the above illustrated student problem, where the response variable is “Student will enroll in coaching class or not” and the answer is YES or NO.
2. **Continuous-variable Decision Tree:** Decision tree, which has continuous response variable.

Table 6.6 gives features of decision trees.

Table 6.6 Features of decision trees

Feature	Description
Application Examples	<ol style="list-style-type: none"> 1. Identify the best combination of products and marketing strategies that target specific sets of consumers in a marketing area. 2. Customer behavior analysis, customer retention strategy planning 3. Fraud detection in industries 4. Diagnosis of diseases
Advantages	<ol style="list-style-type: none"> 1. Decision tree output in the form of graphical representation is very easy to understand. 2. Useful in predicting significant response variable. 3. Not influenced by outliers and missing values to a fair degree as compared to other techniques of modeling.

	4. Handles both numerical and categorical variables. 5. Decision tree is a non-parametric method. Thus, decision trees have no assumptions about space distribution and classifier structure.
Disadvantages	1. Over fitting is a problem associated with decision tree models. Setting constraints on model parameters and pruning solve this problem. 2. When the numerical variables are continuous, the decision tree loses information while categorizing the variables in different categories.

6.7.4.1 Evaluating a Decision Tree

Different decision-tree building algorithms are used. Each algorithm aims to search a variable, which gives the maximum information gain or divides the data in the most homogenous way.

For example, split on GPA in the above example suggests the student enrolls in coaching class more than the Gender of the student.

A decision tree exploits various metrics. For example, Gini index, chi-square and Entropy and Information Gain. Metrics find out the best split variables. A decision tree algorithm CART (Classification and Regression Tree) uses Gini index to split the node. The decision tree resulted by CART algorithm is a binary decision tree (each node will have only two child nodes).

The selection of two items from a population at random must be of the same class and the probability for this is 1 if the population is pure.

1. Gini index is used for categorical response variable “Success” (p) or “Failure” (q).
2. Generates binary splits only.
3. Higher the value of Gini, higher the homogeneity.

Steps to calculate Gini for a split.

1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure

$$(p^2 + q^2)$$
2. Calculate Gini for split using weighted Gini score of each node of that split.

The following example explains the computation of Gini score and when the node split will take place on GPA.

EXAMPLE 6.20

- (i) How will the split on Gender compute?
- (ii) How will the split on GPA compute?

SOLUTION

- (i) To find Split on Gender: Calculate,

$$\text{Gini for sub-node Female} = (0.5) \times (0.5) + (0.5) \times (0.5) = 0.50$$

$$\text{Gini for sub-node Male} = (0.33) \times (0.33) + (0.67) \times (0.67) = 0.5578$$

$$\text{Weighted Gini for Split Gender} = (40/100) \times 0.50 + (60/100) \times 0.5578 = 0.53468$$

- (ii) Similarly, for Split on GPA: Calculate,

$$\text{Gini for sub-node } \geq 9.0 = (0.80) \times (0.80) + (0.20) \times (0.20) = 0.68$$

$$\text{Gini for sub-node } < 9.0 = (0.27) \times (0.27) + (0.73) \times (0.73) = 0.6058$$

$$\text{Weighted Gini for Split Class} = (25/100) \times 0.68 + (75/100) \times 0.6058 = 0.62435$$

Gini score for *Split on GPA* is higher than *Split on Gender*. Hence, the node split will take place on GPA.

The next category of algorithm is based on the chi-square. This finds the statistical significance between the differences between sub-nodes and parent node. The calculation is based on the sum of squares of standardized differences between observed and expected frequencies of the response variable.

The algorithm is used for categorical response variable “Success” or “Failure”. The algorithm generates two or more splits.

The generated tree is known as CHAID (Chi-square Automatic Interaction Detector) which detects using the rule that “higher the value of chi-Square, higher is the statistical significance of differences between the sub-node and the Parent node”. Calculation of chi-square of each node uses the term:

$$= \sqrt{\left(\frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}} \right)} \quad (6.29)$$

Steps to calculate chi-square for a split:

- (i) Calculate the chi-square for an individual node.
- (ii) Calculated the chi-square of Split using the sum of all chi-square of success and failure of each node of the split.

The following example explains chi-square calculation.

EXAMPLE 6.1

How will split compute for Gender and GPA using chi-square calculation?

SOLUTION

To find Split on Gender: Calculate,

Node	Enrolled in Coaching	Not Enrolled	Total	Expected to Enroll (40%)	Not Expected (60%)	Chi-square Enroll Coaching	Chi-square Not Enroll Coaching
Female	20	20	40	16	24	1.0	.82
Male	20	40	60	24	36	.82	.67
Total Chi-square:							3.31

To find Split on GPA: Calculate,

Node	Enrolled in Coaching	Not Enrolled	Total	Expected to Enroll (40%)	Not Expected (60%)	Chi-square Enroll Coaching	Chi-square Not Enroll Coaching
≥9.0	20	5	25	10	15	3.16	2.58
<9.0	20	55	75	30	45	1.83	1.49
Total Chi-square:							9.06

Chi-square also suggests the *GPA split* is more significant compare to *Gender* since Chi-square for GPA split is much higher than Chi-square for Gender split.

A category of algorithm is based on entropy computations and information gain. Entropy in thermodynamics is a measure of disorder or randomness of a system. Statistical studies suggest similar concept to characterize the (im) purity of an arbitrary dataset (randomness in dataset). A pure node requires less information to describe it, and an impure node requires more information. If the subset or the dataset is completely homogeneous, then the entropy is zero and if the sample is an equally divided (50% – 50%), it has entropy of one.

The formula for statistical entropy is:

$$H = \sum_{i=1}^n p_i \log_2 p_i \quad (6.30)$$

Here p is probability of success in that node. Entropy is used with categorical response variable. The split is decided when the entropy is lower than the parent node and other splits. The lesser the entropy value, the better split variable it is.

Steps to calculate entropy for a split:

1. Calculate entropy of parent node.
2. Calculate entropy of each individual node of split and calculate the weighted average of all sub-nodes available in split.

The following example explains the use of entropy computations for the dataset used to calculate Gini in Example 6.20.

EXAMPLE 6.22

How will you split and compute entropies for Gender and GPA using entropy calculation? How much is the information gain?

SOLUTION

1. Entropy for the parent node = $-(40/100) \log_2 (40/100) - (60/100) \log_2 (60/100) = 0.971$. The node is impure.
2. Entropy for the female node = $-(20/40) \log_2 (20/40) - (20/40) \log_2 (20/40) = 1$ and for male node, $-(20/60) \log_2 (20/60) - (40/60) \log_2 (40/60) = 0.92$
3. Entropy for split Gender =Weighted entropy of sub-nodes = $(40/100) \times$

$$1.0 + (60/100) \times 0.92$$

$$= 0.95$$

4. Entropy for GPA ≥ 9.0 node = $-(20/25) \log_2 (20/25) - (5/25) \log_2 (5/25) = 0.72$, and for
 $GPA < 9.0, -(20/75) \log_2 (20/75) - (55/75) \log_2 (55/75) = 0.84.$
5. Entropy for split GPA = $(25/100) \times 0.72 + (75/100) \times 0.84 = 0.81$

Entropy for Split on GPA is the lower than entropy for Split on Gender, so the tree will split on GPA. Information gain from entropy as (1- Entropy).

The basics of decision trees and the decision-making process involved to select the best splits in modeling such tree structure is presented in this section. Decision tree can be applied on regression problems as well as classification problems.

6.7.4.2 Decision Trees in R

Multiple packages are available to implement decision tree in R programming, such as ctree and rpart. The rpart programs build classification or regression models using a two-phase procedure. The resultant models can be represented as binary trees.

First phase is the splitting process using a selected data variable. The process continues recursively either until the sub-groups reach a default minimum size or until no improvement can be made. The second phase consists of using cross-validation to trim back the full tree.

Regression analysis, time series analysis and Markov model use statistical techniques in ML algorithms for predictions. A regression algorithm is a supervised ML algorithm. This means that they predict the value for new data (output value) on learning from model dependencies and relationships between the target output and input features.

Six widely used ML Regression analysis algorithms are as follows:

1. Simple Linear Regression Model— Single input variable and multiple input linear regression
2. Multivariate Regression Algorithm

3. Multiple Regression Algorithm
4. Support Vector Machines Algorithm — Use epsilon-insensitivity (margin of tolerance) loss function to solve regression problems
5. Logistic Regression Classifier Algorithm — It uses SGD method (Table 6.8) and is used for classification (Section 6.7.3)
6. LASSO (Least Absolute Selection Shrinkage Operator) Algorithm

6.7.5 Naïve-Bayes Theorem - Naïve Bayes Classifier

Naïve Bayes is a simple classifier. It is the probabilistic and statistical classifier. It is based on Bayes theorem (from Bayesian statistics) with strong (Naïve) independence assumptions and maximum posteriori hypothesis. It is also a supervised learning technique, which uses non-parametric approach (Posteriori means at the back of something. For example, hypothesis). The classifier assumption is that features have strong independences.

The classifier exploits one of the most basic text classification techniques with a variety of applications in email spam detection, personal email sorting and document categorization.

The technique uses the probabilities of every feature belonging to each class to make a prediction. Makes simpler calculation of probabilities by assuming that the probability of a particular feature belonging to a given class value is independent of all other features. This type of assumption is termed as class conditional independence. The probabilities can be looked up in a single scan of the database and stored in a small table. This makes it a fast and space efficient method.

The use of this method is in decider program for the classifier. The training program trains on maximum likelihood which means maximum probability of occurrences.

Naïve Bayes is widely used for text classification. Vectorization of data points need one pass for vectorization and other for the algorithm. Since the method uses vector implicitly (for example, a bag of word as input), a single pass suffices (Pass means the number of times the input vectors are re-examined during training).

In addition, Naïve Bayes classifier requires a small amount of training data to estimate the parameters, viz., means and variances of the variables, which are necessary for classification. However, training needs high overheads and more time.

Section 9.2.1.3 will give details. A word contained (and its occurrences) represents text in a document. The bag-of-words model is commonly used methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier (Wikipedia). Section 9.2.2 will describe Naïve Bayes Analysis in detail.

Bayes theorem describes the conditional probability of an event. The theorem basis is that conditions that might be related to the event. The probability $P(A|B)$ of Event A occurring and Event B has already occurred is given by the formula:

$$P(A|B) = P(A \text{ and } B)/P(B) = P(A \cap B)/P(B) \quad (6.31)$$

Probability of conditions A or B is equal to the probability of both A and B happening divided by Probability of B alone. (\cap is symbol for intersection in set theory. \cup is symbol for union in set theory).

Naïve Bayes classifiers are efficient in terms of CPU and memory consumption. The classifiers are not sensitive to irrelevant features. They have the ability to handle real and discrete data as well as streaming data well. The classifiers can also be trained very quickly. The only disadvantage is that they assume independence of features. Overall, Naïve Bayes outperforms by other classifiers most of the times and is used as a baseline in many researches.

6.7.6 Support Vector Machine Classifier

Support Vector Machine (SVM) is a method in a set of related *supervised learning method* that uses a vector, which has in general, v elements in v -dimensional space. The vector classifies the data points. Following is a brief introduction.

A data point in the space is represented by a vector. A data point represents by (x_1, x_2, \dots, x_n) in n -dimensional space. Consider two-dimensional space, with data points (x_1, x_2) and axes X_1 and X_2 . Each data-point if considered as a vector element has two components, x_1 , and x_2 . (Two sets of words in text analysis). A

hyperplane is a subspace of one dimension less than its ambient space in geometry. If a space is 3-dimensional, then its hyperplanes are the 2-dimensional planes. However, if the space is 2-dimensional, its hyperplanes are 1-dimensional which means lines.

The algorithm finds Support Vector (SV) to maximize boundary distances. Figure 6.18 shows car sales in different showrooms. A star corresponds to Jaguar sales in a year at different showrooms and dot corresponds to Zest sales of Tata Zest model. X1 axes in feature space and X2 is in metric spaces (sales number).

Figure 6.18 shows the separating hyperplanes using three hyperplanes A, B, and C for classification of data points. A hyperplane is equivalent to a line in case of 2-dimensional space.

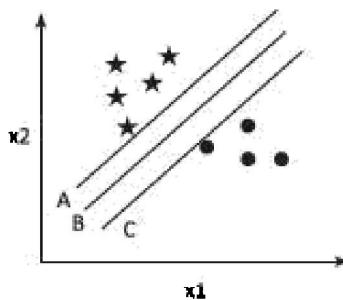


Figure 6.18 Three hyperplanes A, B, and C for classification of data points

The planes are iteratively chosen to maximize distances. SV curve can be specified by not only a plane, but also by a kernel function (such as Gaussian or tri-cube).

Both positive and negative support vectors can be used in features space. Negative SV means the features, which exclude in classification, are used by the classifier.

6.7.7 Random Forest Classifier

Random Forest (RF) uses all four categories of predictor variables. It is an ensemble learning method. Its applications are in classification as well as regression. RF has high overheads. Its architecture is based on a decision tree. The training program trains a regression tree or decision tree (Section 6.7 Figure 6.15).

RF uses a tree learning algorithm modified that selects, such that each candidate splits in the learning process and uses a random subset of features for learning (to take decisions).

For example, Let $\mathbf{I}(\mathbf{T})$ = Training input features or values and $\mathbf{O}[\mathbf{I}(\mathbf{T})]$ = Output exemplary features or values of \mathbf{I} . RF selects a bootstrap sample, randomly from input and output examples (\mathbf{T}). Decision or regression tree \mathbf{D} trains on the sample $s = 1$ to S of $d_s(I_s, O_s)$.

Training algorithm uses bootstrap aggregating method. It puts tree learners in bags (containers). Training dataset input vectors and corresponding output variables bootstrap the sample repeatedly, and fits with the decision tree. The prediction variable can either be estimated by vote (frequency of correct decisions or regression) or by using the averaging formula, in Equation (6.27).

$D(S) = \text{Maximum vote frequency } (s)/\text{Total sample features or}$

$$D(S) = S^{-1} \left[\sum_{s=1}^S d_s(y) \right] \quad (6.32)$$

The advantage of RF is that it effectively programs the conditional relationships and non-linear functions, kernel or other complex functions. Its applications are when the data points are less than 10 M. It has parallel execution with shared nothing architecture.

6.7.8 AdaBoost and Other Ensemble Classifiers

AdaBoost means adaptive boosting. It builds a strong learner from a linear combination of weak learners.

It initially assumes uniform weight of training examples. Weight means importance of an example with respect to other examples. Assume a trainer algorithm using a sample (example) s in trained vector \mathbf{T} .

The $T_s: e \rightarrow \{-1, 1\}$. It means training example T_s weight is -1 or +1 to start with. T_s is a week classifier, as it cannot generate a predictor variable and classify. Each classifier is considered sequentially. When its algorithm increases its weight, then other weights correspondingly reduce.

Now increase the weights of variables those T_e , which are misclassified and need greater importance. Assume weight of T_s is $e(s)$. Finally, the classifier adapts and forms linear combinations of those T_e whose weights were increased.

The equation for Decider D(S) using the linear combination of features is given by:

$$D(S) = K \left[\sum_{s=1}^S \{ \varepsilon(s) \times d_s(y) \} \right]. \quad (6.33)$$

where $K = \sum_{s=1}^S \varepsilon(s)$ is the sum of the weights of each sample, and $d_s(y)$ is the feature of sample s.

OLCs accompanying the book will describe an example of application of AdaBoost.

Self-Assessment Exercise linked to LO 6.6

1. List the meaning of the terms, training example, input vector, predictor variable, continuous variable, categorical variable and feature hashing.
2. What is vectorization? How does a vector differ from a bag of words?
3. How does decision tree algorithm CART (Classification and Regression Tree) use the Gini index to split a node. How is the entropy calculated?
4. When will you use Naïve Bayes, Random Forest and Support vector Classifiers?
5. List features of the Naïve Bayes classifier.
6. How do parallel computations with shared nothing architectures help in efficient Big Data analysis? Which are the classifiers supporting these features?
7. How does AdaBoost generate a strong classifier?
8. What are the limitations of AdaBoost and Random Forest classifiers?

6.8 | RECOMMENDATION SYSTEM

Recommender refers to a machine learning (ML) tool that enables recommendations after extractions of features in the itemsets from multiple datasets. Example is a product recommender, such as book recommender. Recommendation of a recommender enables the selection of the right product among the high recommendation products. A recommender's need arises because a customer finds it difficult to search the right product due to

information overload. Figure 6.19 shows the steps in user-based collaborative filtering (CF) and content-based filtering (CBF) recommenders.

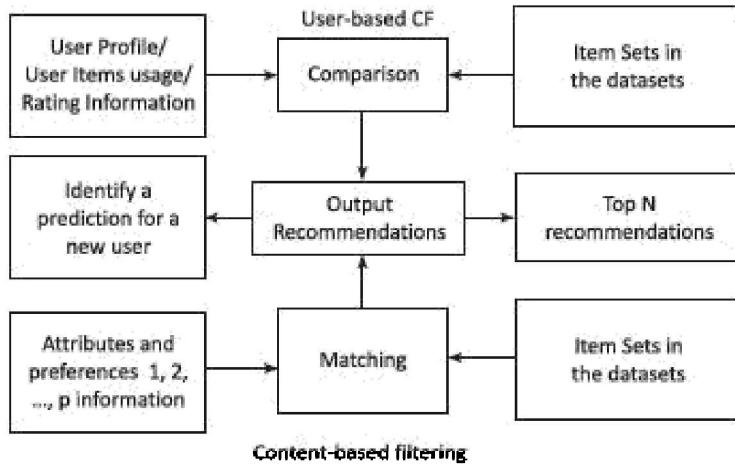


Figure 6.19 Steps in user-based CF and CBF recommenders

6.8.1 Collaborative Recommendation

A recommender generates by collaborative filtering (CF) and gives either prediction or recommendation (Figure 6.19). Prediction is a numerical value (R_{ij}), suggesting the predicted score of item j for the user i . Recommendation can be a list of top-N items that the user will like the most.

Group forms for similar users which means in neighbourhoods. Similar means not distant. The recommender is an application of near-neighbour search method. The already rated items are significant in searching for a neighbour. Once a neighbour of a user is found, different algorithms can be used to combine the preference of neighbours to generate recommendations.

6.8.1.1 Collaborative Filtering

CF algorithm makes recommendations by calculating the similarities in itemsets in-between different items in the datasets. The algorithm predicts the likeliness of an item that it has not rated based on a set of historical preference judgments from a community of users.

CF is different from content-based filtering (CBF) which is built around the attributes of a given item (Figure 6.19). Item features do not capture everything. User's interests may change. CF on the other hand relies on the behavior of the

users.

Top-N Recommendations A step in a CF system is to find Top-N Recommendations (Figure 6.19). Top-N recommendation is used to recommend a set of N top-ranked items that will be of interest to a certain user. For example, a user of a website recommends a list of books (or other products) that may be of his/her interest before leaving the website. Top-N recommendation techniques analyze the user-item matrix to discover relations between different users or items.

The two approaches to collaborative filtering are:

- Model-based collaborative filtering – Based on ML techniques (Section 6.8.2)
- Memory-based collaborative filtering – Based on similarity between users and items

Memory-based Collaborative Filtering Memory-based techniques are best suited to real-life applications due to their effectiveness. Memory-based collaborative filtering can be achieved through two approaches, i.e., one based on items, and the other on users, and termed as user-based and item-based approaches.

6.8.1.2 User-Based Top-N Recommendation Algorithm

An algorithm does the following:

Identify the k most similar users (nearest neighbours) to the active user using the *Pearson correlation* or vector space model. Every user is treated as a vector in the v-dimensional item space. The similarities between the active user and other users are computed between the vectors (Sections 6.3.6 and 6.4.4).

- The corresponding rows are aggregated in the user-item matrix, R to identify a set of items, C, purchased by the group together with their frequencies.
- Recommend the top-N most frequent items in set C that the active user has not purchased.

Pearson correlation model: (i) Does not consider overlapping preferences of two users, (ii) Computation of only one item in case of two user overlapping

preferences, (iii) Correlation does not give added weights when a series of preference values are equal. Pearson correlation using weights is an option to consider these issues.

User-based CF recommends items by finding similar users.

- Users who were interested in the past, are likely to agree again.
- Exploit the opinion of similar users to predict a user's opinion for an item
- Similarity between users is calculated by looking at their overlap in opinions for other items.

User-based nearest-neighbour CF Algorithm, performs the following steps:

- Calculate the similarity or weight, $w_{i,j}$, which reflects distance, correlation or weight, between two users or two items, i and j
- Identify a prediction for the new user by taking the weighted average of all the ratings of the user, item on a certain item or user, or using a simple weighted average
- Suggestions for Top-N recommendation
- The task to generate a top-N recommendation requires finding k most similar users or items (nearest neighbours) after computing the similarities
- Aggregate the neighbours to get the top-N most frequent items as the recommendation.

6.8.1.3 Item-based Collaborative Filtering

An item-based CF recommends items by finding similar items. Amazon initially developed the item-based method. The method calculates similarity between items and makes recommendations. Different items that are purchased together are involved to draw inferences about the relationship between items. The more often two items (say, chocolate, gift box and greeting card) appear in the same shopping cart or user history, they are considered as associated with one another. Association between two items specifies by the observations, such as when a customer adds a greeting card to the purchase cart, the algorithm will suggest things that are associated with items in the cart like chocolate and gift box, over things that are not associated, such as electronic items.

- Items that are purchased together in the past are likely to be selected again.
- Similarity between items is decided by looking at their overlap in the purchase pattern for items.

Exploit the full or a sample of the user-item database to generate a prediction. Users part of a group show similar interest. A prediction of preferences on new items for a new user (or active user) can be evaluated by identifying the neighbours.

Problems with User-based Collaborative Filtering Following are some problems associated with using user-based collaborative filtering:

- User Cold-Start Problem: The technique of filtering is based on user history, but what if the user history is not available? This is referred as the “cold start” problem, and it can apply both to new items and to new users. Not enough information exists about a new user to decide the similarity.
- Sparsity: Users have rated only a few items when recommending from a large dataset of itemsets. This makes it hard to find similar users.
- Scalability: For millions of ratings, computations become slow.

User-based vs Item-based Collaborative Filtering User-based similarity between users is dynamic. Pre-computing the user neighbourhood can lead to poor predictions. User similarities have a much vast domain. Item-based similarity between items is more static, and hence it enables pre-computation which improves online performance. An item-based similarity is more meaningful.

Let a user assign a rating to an item, such as a book on a topic. The similarity between two users of the items can be measured by treating each user as a vector of rating frequencies and computing the cosine of the angle formed by the frequency vectors. The basis of recommendation is similarity. (Vector cosine similarity between two sets of data points (vectors A and B) is given by Equation (6.23b)).

Formally, if P is an $m \times n$ user-item matrix, then the similarity between two items j and i is defined as the cosine of the angles between v -dimensional vectors corresponding to the i -th and j -th columns of matrix P .

Figure 6.20 shows prediction on an item and top 10 recommendation list for a user.

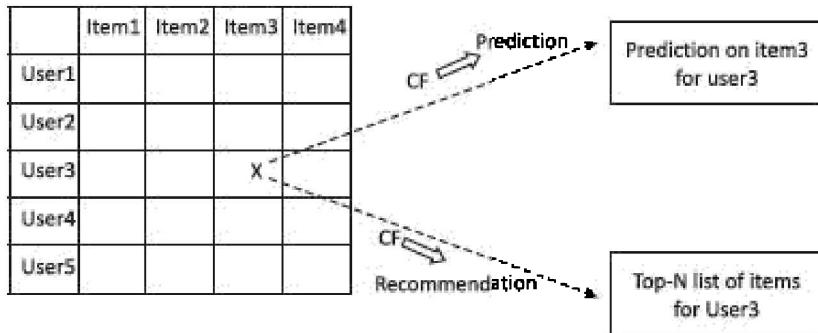


Figure 6.20 Prediction on an item and top 10 recommendation list for a user

6.8.1.4 Computations of Prediction and Recommendation

Next step in a collaborative filtering system is to obtain predictions or recommendations. A subset of nearest neighbours of the active user is chosen based on their similarities in the neighbourhood-based CF algorithm. A weighted aggregate of their ratings is used to generate predictions for the active user.

Weighted Sum of Others' Ratings Calculate a weighted average of all the ratings on a certain item i to make a prediction for the active user according to the following formula:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}, \quad (6.34)$$

where \bar{r}_a and \bar{r}_u are the average ratings for the user a , and user u on all other rated items, and $w_{a,u}$

is the weight between the user a and user u . The summations are overall the users $u \in U$ who have rated the item i .

Simple Weighted Average Use the simple weighted average to predict the rating $P_{u,i}$ for user u on item i in case of item-based prediction.

Item-based Top-N Recommendation Algorithms These algorithms consider the scalability problem of user-based top-N recommendation algorithms.

- Compute k most similar items for each item according to the similarities.
- Identify the set C as candidates of recommended items by taking the union of the k most similar items and removing each of the items in the set U that the user has already purchased.
- Calculate the similarities between each item of the set C and the set U.
- The resulting set of the items in C, sorted in decreasing order of the similarity, will be the recommended item-based Top-N list.

The combined distribution of a set of items may result into different from the distributions of the individual items in the set. The method can suggest sub-optimal recommendations in such cases. Higher-order item-based top-N recommendation algorithms can be used in such situations to consider all combinations of items.

6.8.2 Model for Recommendation Systems

Figure 6.15 showed model building approach for building classifier using machine learning. Similarly, a model building for recommender approach utilizes machine learning. They can quickly recommend a set of items since they use pre-computed models. They use the association rules, regression, clustering, neural network, Bayesian classifiers, decision trees and matrix completion techniques.

Matrix completion means filling the missing entries in sparse matrices. The recommender uses user-item matrices. The matrix completion algorithm uses an algorithm similar to principal component analysis (Section 6.9). A model-based technique analyzes the user-item matrix to identify relations between items. These relations are used to compare the list of top-N recommendations. Model-based techniques resolve the sparsity of data problems associated with recommendation systems.

A model-based recommender is an association-rule-based technique. The following example explains how the association rule (Section 6.5.2) applies in designing a recommender system for books.

How does association rule help in designing a recommender system for books?

SOLUTION

An online bookstore posts recommendations for buying books and suggests the offers after a buyer makes a purchase. For example, if a buyer selects the book ‘Data Analytics’, a list of related books, such as Data Mining, Statistical Concepts, Machine Learning, Big Data Analytics will be offered as recommendations for future purchase. The association rules suggests that when a book on data analytics is purchased, 25% of the times a book on Big Data analytics, 20% of the times a book on data mining, 20% of the times a book on statistical concepts, and 20% of the time a book on machine learning is bought along with it.

Association rules can also be used to plan market strategies for the store. For example, the book on Big Data analytics can be promoted for the sales of the other three books with or without a discount.

Websites and services such as Amazon, Facebook, YouTube and Google use association rules mining. The recommender looks at patterns of activities between different users and different products to produce the recommendations.

1. Online bookstore: ‘Customer who bought this, also bought’.
2. Online shopping site: ‘You may like this’.
3. Social web: Recommended applications, such as ‘Jobs you may be interested’ in....
4. Search engine: Similar advertising

6.8.3 Content Based Recommendation

Content-based filtering is built around the attributes and preferences of a given item. The recommender evaluates the contents or items on the preferences of others with a similar point of view. When anyone buys a product online, most websites automatically recommend other products that she/he may be interested to buy. Figure 6.19 illustrated these steps.

6.8.4 Knowledge-based Recommendation

A Knowledge-based Filtering (KBF) recommender builds on explicit knowledge about user preferences, the number of items of specific attributes and a criterion function for the recommendation means positive recommendation criterion for the given context.

Advantage of KBF recommender is that it applies in scenarios where CF and CBF have difficulties in uses. For example, cold start difficulty. Another advantage is applicability in complex domains, such as purchase of house, where ratings are mostly unavailable. Another advantage is applicability when using the conversations with experts.

A disadvantage is problem associated with acquiring difficulties for knowledge. A definitive recommendation in explicit form is difficult without full knowledge.

6.8.5 Hybrid Recommendation Approaches

Hybrid filtering is a combination of collaborative and content base filtering for recommendations. CF and CBF have following issues in making recommendations to new users.

CF Issues Three CF issues are (i) sparsity issue (Section 6.8.1.3), (ii) early rater issues, such as that several times a proper rating can be true if rendered after longer usages, and (iii) preference of a group of users does not account for low ratings by certain users.

CBF Issues Three CBF issues are (i) description of contents difficulties in format accessible to a new user, (ii) over-special need of new user, making recommendation outside the specialized information available, and (iii) difficulty in comparing subjective domains information.

Hybrid approach combines CF and CBF. The approach combines both types of information, applying both filters, CF and CBF.

Self-Assessment Exercise linked to LO 6.7

1. How does a recommender provide top 10 recommendations?
2. List the difference in approaches of recommender-based on user-based

collaborative filtering (CF) and content-based filtering (CBF).

3. When is collaborative filtering used?
4. How does a model-based recommender use the association rule?
5. Compare knowledge-based filtering and content-based filtering.

6.9 | APACHE MAHOUT MACHINE-LEARNING APPLICATIONS

Big Data requires fast and efficient processing of very large datasets at the cluster of machines. Big Data analysis uses datasets with over 1 million data points. Mahout has high efficiency for above 10 M data points in shared-nothing environment. Mahout in sequential shared environment has higher time efficiency for less than 1 M data points.

LO 6.8

Apache Mahout architecture and components and their applications for clustering, classification, collaborative-filtering and recommender system

Mahout computational tasks execute fast when using multiple machines as well as multi-core processing units, distributed over the cloud, tasks runs parallel, and run in shared nothing-computational environment.

Mahout requires installing JVM and integrated development environment (IDE). For example, Eclipse, installing Apache Maven and then the Mahout. (Maven in English means a person with good knowledge or understanding of a subject. Apache *Maven* is a build automation tool used primarily for Java projects).

Mahout is a scalable generalized tensor and linear algebra solving engine. Mahout vectors specify in three Java Classes: SequentialAccessSparseVector, RandomAccessSparseVector, and DenseVector (Section 6.4.4.4). Sequential access sparse means accessing two parallel vectors, one of keys and other of values. Random access means accessing vectors using key, index or hash followed by values, in any order.

Mahout is a tool from Apache Foundation that runs the ML algorithms on Big Data in a parallel computing environment. Mahout primarily focuses on clustering and classification, collaborative filtering and regression analysis.

Mahout consists of tools to automate finding of meaningful patterns at big datasets stored in data store at HDFS.

Features of Mahout are as follows:

1. Mahout is designed on top of Apache Hadoop, using the supported algebraic platforms like fast computing Apache Spark paradigm and MapReduce (Spark is fast compared to MapReduce).
2. Offers effective and faster algorithms to analyze large datasets.
3. Contains several Spark and MapReduce enabled clustering implementations, such as k-means, fuzzy k-means, Canopy, Dirichlet and mean shift.
4. Supports distributed Naïve Bayes and complementary Naïve Bayes classification implementations,
5. Designed for a distributed computing environment, but includes contributions that run on a single node or on a non-Hadoop cluster also, such as ‘Taste’ collaborative-filtering recommender,
6. Mahout, besides collaborative filtering baser recommender, includes other recommenders also, say (i) SVD recommender (ii) KNN-item-based recommender (linear interpolation item based recommender), (iii) cluster-based recommender.
7. Exploits Apache Hadoop library to scale effectively in the cloud.
8. Includes APIs for distributed and in-core first and second moment routines, distributed row matrix (DRM), distributed and scalable libraries for matrix and vector, distributed and local principal component analysis (DSPCA and SPCA)and stochastic singular value decomposition (DSSVD and SSVD), singular value decomposition (SVD), Distributed Cholesky QR (thinQR), Distributed regularized Alternating Least Squares (DALS), Java libraries for common mathematics and statistical operations (focused on linear algebra) and primitive Java Collection Interfaces.
9. Provides an easy to use framework for processing large volume of data, hence is suitable in big data environment.

(*Principal Component Analysis (PCA)* means a linear transformation method for finding the directions of maximum variance in high-dimensional data and project those for transformation onto a smaller dimensional subspace while retaining most of the information. PCA identifies patterns in data sets and detect the correlation among the variables. When there is a strong correlation, then try for reducing the dimensionality. PCA applies in a number of use cases, such as stock market predictions, and the analysis of gene expression data.)

Mahout 0.13.0 released on 17 April 2017 enables easier implementations of most modern machine learning and deep learning algorithms. The versions include the open-source distributed scalable linear algebra library ViennaCL, the Java wrapper library interface JavaCPP and the graphics processor manufacturer, NVIDIA CUDA bindings directly into Mahout. The new version of Mahout makes it easier to run matrix mathematics on graphics cards (used in computers for fast graphic computations). Future versions of Mahout will include support for inclusion of native iterative solvers, according to Apache.

Mahout is used by big companies, such as Adobe, Facebook, LinkedIn, Foursquare, Twitter and Yahoo. The companies use Mahout for the following:

- (i) Recommendation engine on Foursquare— Foursquare helps in finding out places, food, and entertainment available in a particular area
- (ii) Pattern mining on Yahoo! as anti-spam
- (iii) Research Gate, the professional network for scientists and researchers, uses Mahout's recommendation algorithms
- (iv) Twitter uses Mahout's Latent-Dirichlet-Allocation (LDA) implementation for user interest modeling
- (v) ADOBE AMP uses Mahout's clustering algorithms to increase video consumption by better user targeting.

Figure 6.21 shows Mahout architecture.

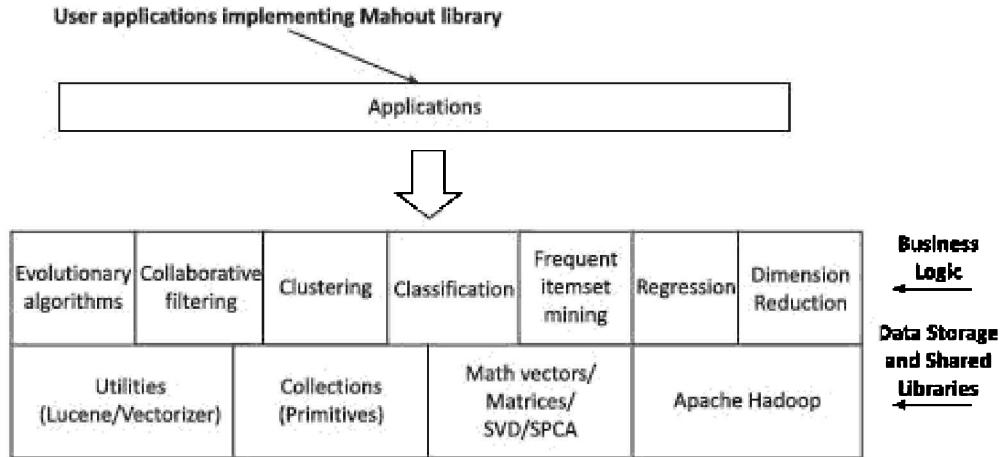


Figure 6.21 Mahout architecture

Mahout implements ML methods as:

1. **Collaborative Filtering** – Enables making automatic predictions about interests of a user. The methods consider the preferences from many users to make predictions (collaborating). Collaborative filtering methods are used by recommender systems and similar itemsets mining. For example, user-based, weighted matrix factorization SVD++, and parallel SGD (in sequentially in shared-data environment), item-based, matrix factorization with alternating least squares, matrix factorization with alternating least squares with implicit feedback, using parallel scalable shared nothing (MapReduce) as well as sequential algorithms. Some popular websites that make use of the collaborative filtering include Amazon, Netflix and iTunes.
2. **Clustering** – Take items in a particular class and organize them into groups, such that items belonging to the same group are similar to each other. Mahout includes the ML algorithms for k-means, fuzzy k-means, Canopy, Dirichlet, and mean-shift for the clusters analysis (Section 6.9).
3. **Classification** – Learns from existing categorizations and then assigns unclassified items to the best category. For example, Naïve Bayes and Random Forest using parallel scalable algorithm (parallel shared-nothing) and logistic regression, support vector, Hidden Markov model and multi-layer perceptron (sequentially shared data environment).

4. **Frequent Itemset Mining** – Analyses items in a group and identifies which items typically appear together.

Clustering algorithms are canopy, k-means, fuzzy k-means, streaming k-means and spectral clustering, using parallel scalable (MapReduce) as well as sequential algorithms. Table 6.7 gives brief descriptions of them.

Table 6.7 Clustering methods

Canopy clustering	Pre-processes the data before using a k-means or hierarchical clustering algorithm
K-means clustering	Deduces partition of n observations into k clusters in which each observation belongs to the cluster with the nearest mean
Mean shift clustering	Retrieves modes or clusters in 2-dimensional space, where the number of clusters is unknown
Fuzzy k-means	Discovers soft clusters where a particular point can fall into more than one cluster
Hierarchical clustering	Builds a hierarchy of clusters using either an <i>agglomerative</i> (bottom-up) or <i>divisive</i> (top-down) approach
Spectral clustering	Finds cluster points using Eigenvectors of matrices derived from the data
MinHash clustering	Estimates similarity between two datasets quickly
Dirichlet process clustering	Performs Bayesian mixture modeling
Latent Dirichlet Allocation	Automatically and jointly cluster words into <i>topics</i> and <i>documents</i> into mixtures of topics

Three other categories of Mahout algorithm belong to the collaborative filtering, classification or frequent itemset mining. Mahout provides two collaborative-filtering algorithms:

1. One is distributed item-based collaborative filtering. The method estimates a user's preference for an item based on the preferences for similar items

and row matrices.

2. Other is collaborative filtering using a parallel matrix factorization. The method finds the items the user might prefer among a matrix of items which a user has not seen so far.

Table 6.8 Brief descriptions of collaboration filtering, Classification and frequent itemset Hadoop-compatible algorithms

Algorithm	Description
Collaborative Filtering	
Distributed item-based collaborative filtering	Estimates a user's preference for an item on the basis of the preferences for similar items and row matrices
Collaborative filtering using a parallel matrix factorization	Finding the items, the user might prefer among a matrix of items, which a user has not seen so far.
Classification	
Bayesian	Classifies objects into binary categories; Naïve Bayes classification.
Random Forests	Provides a collective learning method for classification (and regression) that operate by constructing a multitude of decision trees
Stochastic Gradient Descent (SGD)	Iterative learning algorithm in which each training example is used to pull the model slightly to give more closer to correct answer for that one example (Logistic regression algorithm uses the SGD)
Frequent Itemset Mining	
Parallel frequent pattern growth algorithm	Analyzes items in a group and then identifies which items typically appear together

Mahout Recommender Engine Mahout provides recommender engines of

several types, such as (i) user-based recommenders, (ii) item-based recommenders, and (iii) several other algorithms. Mahout has a non-distributed, non-Hadoop-based recommender engine. A text document is the input and has user preferences for items. The output of this engine would be the estimated preferences of a particular user for other items. Five components in Mahout to build a recommender engine are as follows:

1. Data Model– The object of this class holds a file that contains the users, items and preferences details of a product.
2. User Similarity– A measure that returns a number representing how similar the given two users are.
3. Item Similarity– Defines a notion of similarity between two items.
4. User Neighbourhood– For finding the neighbourhood of a given user.
5. Recommender– Takes data model, neighbourhood and user similarity together to produce recommendations.

A data model is prepared from large datasets and is passed as an input to the recommender engine. The recommender engine generates the recommendations for a particular user.

Self-Assessment Exercise linked to LO 6.8

1. Why does Mahout compute faster in case of data points above 1M compared to sequential non-scalable programming?
2. List the features of Mahout 0.13 version and expected new version features.
3. List and differentiate between two Mahout collaborative-filtering algorithms, collaborative filtering using a parallel matrix factorization and distributed item-based collaborative filtering.
4. Write meanings of distributed row matrix (DRM), distributed and scalable libraries for matrix and vector, distributed and local principal component analysis (DSPCA and SPCA) and stochastic singular value decomposition (DSSVD and SSVD), singular value decomposition (SVD).



KEY CONCEPTS

AdaBoost
ANOVA
Apache Mahout
Apriori algorithm
Artificial intelligence
association rules
Bayesian classification
category variable
candidate rules
chi-square
classification
clustering
collaborative filtering
confidence level
correlation
cosine similarity
data mining
decision tree
dimension
distance measure
distribution function
edit distance
entropy
Euclidean distance
explanatory variable
F-test

feature variable
frequent itemset
Gini index
hypothesis
interaction variable
Jaccard similarity
K-mean
K-NN
kernel function
Manhattan distance
market basket model
moment
multiple regression
multivariate regression
Naïve Bayes classifier
non-linear transformation
null hypothesis
objective function
outcome variable
outlier
Pearson correlation
population
predictor variable
probability distribution function
Random Forest
recommender
regression model
relationship

response variable
sample
scatter plot
similar itemsets
similarity
similarity coefficient
singular value decomposition
standard deviation
standard error of estimate
statistical inference
statistical significance
stochastic gradient descent
supervised learning
support vector
test dataset
training data
unequal variance
unsupervised learning
user neighbourhood
variable
variance
Welch's t-test



Learning Outcomes

LO 6.1

1. Mathematical and statistical methods estimate the relationships, outliers, variances, probability distribution and correlations in variables, items or entities. Scatter plot depicts the relationship between two variables, and suggests whether they have linear or non-linear relationship.
2. Variance means dispersion with respect to the expected. *Moments* (0, 1, 2, ...) refer to expected values to the powers of (0, 1, 2, ...) of random-variable variance.
3. Correlation is a statistical technique that is used to measure and describe the strength and direction of the relationship between two variables.

LO 6.2

1. Linear and non-linear regression model-based analysis predicts the values of one variable, given the values of another variable. More than one variable can be used as predictor with multiple regressions.
3. Regression analysis is a powerful technique used for predicting the unknown value of a variable from the known value of another variable.
4. K-NN method uses Euclidean, Manhattan, Hamming and other distance measures for regression analysis. K-NN predicts using interpolation, extrapolation and averaging methods using weights.

LO 6.3

1. Methods of finding similar items and similarities are nearest neighbour search, Jaccard similarity and collaborative filtering.
2. The similar items and similarities use the distance measures. The distance measures are Euclidean, Jaccard, cosine, edit and Hamming distances.

LO 6.4

1. Frequent Itemset Mining (FIM) is one of the popular techniques to extract knowledge from the data. The technique has been an essential part of data analysis and mining. The extraction is based on frequently occurring

events.

2. Market basket analysis is a tool of knowledge discovery about co-occurrence of items. A co-occurrence means two or more things happen together. It can also be defined as a data mining technique to derive strength of association between a pair of product items.
3. Objective of generation of association rules is to find uncovered relationships using some strong rules.
4. Apriori algorithm is used for frequent itemset mining and association rule mining. Apriori algorithm simply follows a basis that any subset of a large itemset must be a large itemset. This is called the Apriori principle, and can reduce the number of itemsets which an algorithm needs to examine. Apriori principle suggests, if an itemset is frequent, then all of its subsets must also be frequent.

LO 6.5

1. Cluster analysis means finding the grouping of the objects (datasets) of similar types or characteristics. ML algorithms methods are K-means, K-medoids, Fuzzy K-means, Canopy, and Dirichlet for clusters analysis.
2. A clustering algorithm finds k clusters in a given dataset using k centroids.

LO 6.6

1. Methods of machine learning are supervised and unsupervised learning. Clustering and classification differ. Clustering is identification of groups of similar objects. Classification means splitting the datasets into subsets with similar features using statistical concepts.
2. Classifiers use training datasets, input vectors, output vectors and predictor variables. Classifier algorithm components are training, model and decider programs.
3. Classifiers, for example Random Forest, use decision tree algorithms which evaluate decision tree.

4. Naïve Bayes is a simple, probabilistic and statistical classifier. Naïve Bayes classifier basis is Bayes' Theorem (from Bayesian statistics) with strong (Naïve) independence assumptions and maximum posteriori hypothesis. The classifier uses are in text and documents.
5. Classification methods are k-nearest neighbour, Stochastic Gradient Descent - Logistic Regression, Support Vector Machine, Random Forest and AdaBoost classifiers.

LO 6.7

1. Recommender system is a system that evaluates contents or items on the preferences of others with a similar point of view. The two approaches to collaborative filtering are (i) memory-based collaborative filtering – based on similarity between users and items, and (ii) model-based collaborative filtering – based on ML techniques.
2. Collaborative filtering predicts the likeliness of an item that she/he has not rated based on a set of historical preference judgments from a community of users.
3. Collaborative filtering builds around the attributes of a given item. Collaborative filtering relies on the behavior of the users.

LO 6.8

1. Apache Mahout is scalable generalized tensor and linear algebra solving engine, which runs the ML algorithms on Big Data in parallel computing environment. Mahout enables clustering, classification, collaborative filtering, regression and recommender. Mahout consists of tools to automate finding of meaningful patterns at big datasets stored in data store at HDFS.
2. Contains several Spark and MapReduce enabled clustering implementations, such as K-means, Fuzzy K-means, Canopy, Dirichlet, supports distributed Naïve Bayes and complementary Naïve Bayes classification implementations.

Objective Type Questions

Select one correct answer option for each of the following questions:

6.1 Mahout includes APIs for (i) distributed and in-core first and second moment routines,

(ii) distributed row matrix (DRM), (iii) distributed and scalable libraries for tensor, matrix and vector, (iv) distributed and local principal component analysis and stochastic singular value decomposition, (v) distributed Cholesky QR (thinQR), (vi) distributed regularized alternating least squares, (vii) Java libraries for common mathematics and statistical operations, (viii) focuses on linear algebra for matrices and vectors, and (ix) primitive C++ and well as Java collections

- (a) i to viii
- (b) all
- (c) all except i and v
- (d) i to v, viii and ix

6.2 (i) Regression analysis is a technique used for predicting, (ii) predicts the unknown value of a variable from the known value of another variable. (iii) Regression analysis is a statistical method. (iv) Regression deals with the formulation of conceptual model depicting a relationship amongst dependent and independent variables. (v) The independent variable is used for the purpose of prediction of the values. (vi) More than one variable whose values are hypothesized are called independent variables. (vii) The prediction for the dependent variable can be made by accurate selection of the independent variables to estimate a dependent variable.

- (a) none
- (b) all
- (c) all except v and vi
- (d) iii to vii

6.3 (i) The standard error of the estimate is a measure of the dispersion (or

variability) in the

(ii) predicted values in a regression, (iii) probabilistic values in a regression. (iv) When the s_{est} is small, most of the observed values (y) dots are close to the regression line in a scatter plot and better is the estimate based on the equation of the line. (v) When the s_{est} is small, many of the observed values are far away from the regression line. (vi) When the standard error is zero, then no variation exists corresponding to the computed line. The correlation is perfect.

- (a) iv
- (b) all
- (c) all except ii to v
- (d) all except iii and v

6.4 (i) Coefficient values suggest which relationships in the model are statistically significant and (ii) the p-values in regression analysis suggest the nature of those relationships. (iii) The coefficients describe the mathematical relationship between each independent variable and the dependent variable. (iv) The p-values for the coefficients indicate whether these relationships are statistically significant.

- (a) all except ii
- (b) iii and iv
- (c) all
- (d) ii to iv

6.5 A hypothesis test requires (i) stating the hypotheses, (ii) null hypothesis, (iii) alternative hypothesis, (iv) F-test hypothesis, and (v) t-test hypothesis. Also required is (vi) preparing plan for the analysis. (vii) The analysis plan means how to use sample data to accept or reject the alternative hypothesis.

- (a) i, iii, iv and vi
- (b) i to v

- (c) ii to vi
- (d) i, ii, iii, vi

6.6 K-mean based algorithm (i) classifies or groups the objects based on the attributes (features) into k number of groups. k is a positive integer number. (ii) The grouping of data results into k clusters (C_1, C_2, \dots, C_K), each has A centroid. (iii) Centroid is fundamentally a center representative of a cluster. (iv) The centroid of each cluster is calculated as the mean of all the instances belonging to that cluster. (v) The clusters are formed by maximizing the sum of squares of distances between the data and the corresponding cluster centroid.

- (a) all except v
- (b) i, ii, iii, v
- (c) ii to v
- (d) all except iii

6.7 (i) Clustering *finds* to which class a new object belongs to from the set of predefined classes.
(ii) Classification *groups* a set of objects in order to find the relationship between them. Clustering and classification *differ* in terms of (iii) supervised and unsupervised learning,
(iv) use of training datasets, no training datasets usage (v) labels (all the data labeled, unlabeled), (vi) datasets consisting of attributes and class labels, (vii) process algorithms (categorization of new data according to the observations of the training set, usages of statistical concepts and splitting of datasets sub-sets with similar features), respectively.

- (a) all except i
- (b) all except iii and iv
- (c) all except i, iii
- (d) all

6.8 Formal statement of the association rule problem is stated as: Let $I = \{I_1, I_2,$

..., I_d } be a set of d distinct attributes, also called literals. Let $T = \{t_1, t_2, \dots, t_n\}$ be set of transactions and (i) contains a set of items such that $T \subseteq I$. (ii) An association rule is an implication of the form $X \rightarrow Y$,
(iii) where X, Y belongs to sets of items called itemsets ($X, Y \subseteq I$), and (iv)
 X and Y are union of itemsets ($X \cup Y = I$). Here, (v) X is called consequent,
and (vi) Y antecedent.

- (a) i, ii and v
- (b) all except i, v
- (c) i to iii
- (d) i to iv

6.9 Apriori algorithm is simple as (i) it follows a basis that any subset of a large itemset must be a large itemset, called Apriori principle. (ii) The Apriori principle can reduce the number of itemsets need to examine. (iii) Apriori principle suggests if itemset {A, B, C} is a frequent itemset, then all its subsets {A}, {B}, {C}, {A, B}, {B, C} and {A, C} need not be frequent. (iv) On contrary, if an itemset is not frequent, then none of its supersets can be frequent. (v) Apriori advantage is that the principle results in to a smaller list of potential frequent itemsets as mining progresses. (vi) The algorithm requires multiple scans of the database. (vii) The complex generation process for candidate exploits more time, space and memory.

- (a) i to v
- (b) all but vi and vii
- (c) all except iii
- (d) i to iv

6.10 (i) A recommender system is a system that evaluates contents or items on the preferences of others with a similar point of view. (ii) Collaborative filtering predicts the likeliness of an item that he/she has not rated based on a set of historical preference judgments from a community of users. (iii) Collaborative filtering is similar from content-based filtering, (iv) which is built around the attributes of a given item. (v) Item features capture

everything. When user's interest changes then also the collaborative filtering does not on the other hand rely on the behavior of the users.

- (a) i, ii, iv
- (b) all except v
- (c) all except iii
- (d) all except i

6.11 Item-based collaborative filtering uses (i) full or (ii) a sample of the user-item database to (iii) generate a prediction. (iv) Users of a group shows dissimilar interest. (v) A prediction of preferences on new items for a new user (or active user) can be evaluated by identifying the neighbours. (vi) Jaccard Similarity, (vii) cosine similarity, (viii) edit-distance or (ix) correlation methods are used to find similarities between users.

- (a) all except viii
- (b) all except iv
- (c) i to vii
- (d) all except i

6.12 User-based nearest neighbour collaborative filtering algorithm, performs the following steps: (i) Calculate the similarity or weight $w_{i,j}$, which reflects distance, correlation or weight between two users or two items, i and j. (ii) Identify a prediction for the new user by taking the weighted average of all the ratings of the user or item on a certain item or user, or (iii) using variance. (iv) Suggest top-N recommendations. (v) The task to generate a top-N recommendation requires finding k most similar users or items (nearest neighbours) after computing the similarities, and (vi) aggregate the neighbours to get the top N most frequent items as the recommendation.

- (a) all
- (b) i, iii and iv
- (c) all except iv

(d) all except iii

6.13 The decision trees (i) aggregate the datasets based on all values of three variables and identify the variable, which creates the (ii) best heterogeneous sets of datasets (which are (iii) homogeneous to each other). (iv) Decision tree output in the form of (v) graphical representation is very easy to understand. (vi) Useful in predicting significant response variable. (vii) Influenced by outliers and missing values to a fair degree as compared to other techniques of modeling. (viii) Handles both numerical and statistical variables. (ix) Decision tree is considered a non-parametric method. Thus, (x) decision trees have no assumptions about the space distribution and the classifier structure.

(a) all

(b) iv, v, vi, viii, ix, and x

(c) all except i

(d) all except i and x

6.14 Naïve Bayes is (i) simple, (ii) probabilistic and (iii) statistical classifier. (iv) The classifier base is Bayes theorem (from Bayesian statistics) with strong (Naïve) independence assumptions and maximum posteriori hypothesis. (v) It is an unsupervised learning technique, which uses non-parametric approach. (vi) The classifier exploits one of the most basic text classification techniques with a variety of applications in email spam detection, personal email sorting and document categorization. (vii) Classifier assumes class conditional dependence.

(a) all except vi and vii

(b) all except ii

(c) All except v and vii

(d) All except iii and vi

Review Questions

- 6.1 Describe the approaches used in linear, multivariate and multiple regression algorithms. **(LO 6.1)**
- 6.2 How are correlations indicators used for evaluating linear relationships? What do the strength and direction of the relationship describe? **(LO 6.1)**
- 6.3 How do Euclidean and Manhattan distances perform regression analysis and predictions? How do the weights apply? **(LO 6.2)**
- 6.4 How are Jaccard distance, cosine distance and edit distance used for finding similar items? **(LO 6.3)**
- 6.5 How does frequent pattern mining enable knowledge discovery from a large number of itemsets? **(LO 6.4)**
- 6.6 Describe the Apriori algorithm. **(LO 6.4)**
- 6.7 How are methods of collaborative filtering used? **(LO 6.4)**
- 6.8 How does clustering a large dataset help in knowledge discovery about the datasets? **(LO 6.5)**
- 6.9 Compare the characteristics features of K-means, K-medoids, Canopy and hierarchical clustering techniques. **(LO 6.5)**
- 6.10 What are the differences between clustering and classification algorithms, and between supervised and unsupervised learning? Explain functions of training, model and decider programs. **(LO 6.6)**
- 6.11 How is a decision tree used? **(LO 6.6)**
- 6.12 Compare the characteristic features of K-NN, SGD, Naïve Bayes, Random Forest, Support Vector and AdaBoost classifiers. **(LO 6.6)**
- 6.13 What are the steps in item-based collaborative filter when used for recommender? **(LO 6.7)**
- 6.14 List and explain the features of Mahout 0.13. List the machine learning algorithms, which Mahout implements. **(LO 6.8)**
- 6.15 How is principal component analysis used in machine learning algorithms? **(LO 6.8)**
- 6.16 How do algorithms provided in Mahout perform clustering and

classification? (LO 6.8)

Practice Exercises

6.1 Consider student expected grades in an examination. Assume that grades have probabilistic distributions. Consider grade point as a variable. Assume that probability of student awarded grade point 1.0 is 1%, 2.0 is 4%, 3.0 is 8%, 4.0 is 16%, 5.0 is 24%, 6.0 is 26%, 7.0 is 11% and 8.0 is 10%. (i) What are the mean, variance and standard deviation? (ii) Compute the 0th moment, 1st moment, 2nd moment and 3rd moment. (iii) How do moment and variance relate? (iv) Interpret the parameters computed. (LO 6.1)

6.2 Recapitulate Practice Exercise 3.1. Consider a car company selling Jaguar Land Rover, Hexa, Zest, Nexon and Safari Storme models of cars. Table 6.9 gives a dataset for sales of these cars at 2000 showrooms for each date in 2017. Total 2000×365 rows per year. Total columns are $2000 \times 365 \times 7$ per year. Variables for sales are denoted as x followed by day of the year (1 or 2, ..., 365) followed by the model_ID, (1, 2, 3, 4 or 5) in columns 3 to 7. Using interaction variables concept, list the steps to find whether any relationship between the monthly sales figures of car models exists or not.

Table 6.9 Sales data for cars of 5 models at 2000 showrooms for each date in 2017

CCSR_id	Date (DT) mmddyy	Jaguar Land Rover- daily Sale (JLRDS)	Hexa daily Sales (HDS)	Zest- daily Sales (ZDS)	Nexon Weekly Sale (NWS)	Safari Storme Weekly Sale (SSWS)
0001	010117	x11	x12	x13	x14	x15
0001	010217	x21	x22	x23	x24	x25
0001	010317	x31	x32	x33	x34	x35

CCSR_Id	Date (DT) mmddyy	Jaguar Land Rover- daily Sale (JLRDS)	Hexadaily Sales (HDS)	Zest- daily Sales (ZDS)	Nexon Weekly Sale (NWS)	Safari Storme Weekly Sale (SSWS)
0001	010417	x41	x42	x43	x44	x45
.	-
.	-
.	-
2000	122717	x3631	x3632	x3633	x3634	x3635
2000	122817	x3631	x3632	x3633	x3634	x3635
2000	12 2917	x3631	x3632	x3633	x3634	x3635
2000	123017	x3641	x3642	x3643	x3644	x3645
2000	123117	x3651	x3652	x3653	x3654	x3655

(LO 6.1)

6.3 Show a simple linear regression chart for varying weekly sales of cars of a specific model, say Tata Zest all over the country. Plot the values on a graph, with week of the year on the x axis and weekly sales on the y axis. How are variance and moments 0, 1, 2 and 3 estimated? Assume the plot fits a line $\text{Sales at 52nd week} = \text{Sales at 1st week of the year} + 0.001 \times 52 \times \text{Sales at 1st week}$. How can the sales at 100th week be predicted from regression analysis? **(LO 6.2)**

6.4 List the steps to use the linear regression model to find the rate of monthly sales growth of each car model combined at all showrooms. **(LO 6.2)**

6.5 List the steps to prepare a new table. Table 6.10 gives yearly accumulated sales of each model from 2010 to 2017, i.e., total 8 rows for 8 years of accumulated yearly sales and total columns are 8×7 per year. Variables for sales are denoted as s followed by day of the year (1 or 2, ..., 8). Use values in each cell in terms of variable s. Table 6.9 format can be as shown below.

Table 6.10 Yearly sales data for cars of 5 models accumulated over all showrooms and all dates in a year during 2010 to 2017

S.No.	Year yyyy	Jaguar Land Rover daily Sale (JLRYS)	Hexa daily Sales (HYS)	Zest daily Sales (ZYS)	Nexon Weekly Sale (NYS)	Safari Storme Weekly Sale (SSYS)
1	2010	s11	s12	s13	s14	s15
2	2011	s21	s22	s23	s24	s25
3	2012	s31	s32	s33	s34	s35
4	2013	s41	s42	s43	s44	s45
5	2014	-	-	-	-	-
6	2015	-	-	-	-	-
7	2016	-	-	-	-	-
8	2017	s81	s82	s83	s84	s85

(LO 6.2)

- 6.6 List the steps to find regression coefficients in linear model and non-linear model, moments, variance and standard deviation of actual yearly sales vs estimates from the coefficient values for each car model. **(LO 6.2)**
- 6.7 List the steps to find regression coefficients using linear model and Euclidean distances.
(LO 6.2)
- 6.8 List the steps in coding for determining the Jaccard, cosine and Euclidean similarity coefficients. **(LO 6.3)**
- 6.9 List the steps for finding the association, association rules, filtering the frequent itemsets for computer science books on two programming languages, Java and Python. **(LO 6.4)**
- 6.10 Assume a graph with price discount given between 0% to 40 % with respect to total sales realised to that customer. Show the dots for each customer. How will the company optimize the profit using that data and take pricing discount decisions using clustering algorithm. **(LO 6.5)**
- 6.11 Describe hierachial approaches including agglomerative-divisive-distance measures to defining the distance between clusters in different algorithms. **(LO 6.5)**
- 6.12 List the steps in Schoastic Gradient Descent and logistic regression. Give an example of using SGD logistic regression. **(LO 6.6)**

6.13 List the steps in a AdaBoost classifier. Give an example of using SGD logistic AdaBoost.
(LO 6.6)

6.14 List the steps for using the hidden Markov deep learning algorithm. Give an example of using the hidden Markov. **(LO 6.6)**

6.15 List the steps for using the multi-level perceptron deep learning algorithm. Give an example of using multi-level perceptron. **(LO 6.6)**

6.16 List the steps for cluster analysis and find whether clustering shows higher sales in certain group of showrooms in festive months, i.e., Octobers and Decembers in most of years compared to other months. **(LO 6.6)**

6.17 List the steps for the decision trees from the data in Table 6.9. **(LO 6.6)**

6.18 List the steps for collaborative recommendation, and mining the similar itemsets in the problem in Practice Exercise 6.6. **(LO 6.7)**

6.19 List the coding steps using Mahout 0.13 and Python libraries for the algorithms listed in Table 6.6. **(LO 6.8)**

1 [https://en.wikipedia.org/wiki/Kernel_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics))

2 Weber, Roger; Schek, Hans-J.; Blott, Stephen, ‘A quantitative analysis and performance study for similarity search methods in high dimensional spaces’

3 https://en.wikipedia.org/wiki/Levenshtein_distance

Note:

○○● Level 1 & Level 2 category

○●● Level 3 & Level 4 category

●●● Level 5 & Level 6 category

Chapter 7

Data Stream Mining and Real-Time Analytics Platform—SparkStreaming

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- LO 7.1 Get conceptual understanding of data stream, model, architecture, management, sources, querying and stream processing issues
- LO 7.2 Get knowledge of methods of sampling, filtering and counting distinct elements in stream computing
- LO 7.3 Get knowledge of methods of finding the frequent itemsets in a stream, handling the large datasets and mining the association-rules
- LO 7.4 Get introduced to a real-time analytics platform—Apache SparkStreaming, and applications of real-time analytics to sentiments analysis, and the stock prices analysis

RECALL FROM EARLIER CHAPTERS

Apache[®] Spark[™] is a fast and dynamic compute engine with in-memory processing. Processing requires *read* of instructions and data, and *write* of results. The in-memory data processing results in fast computations. In-

memory *read* and *write* operations take place without any delay as compared to read from disk and write to disk operations. Fast computations also result due to the use of DAGs and acyclic dataflows. Spark processes the data stored at HDFS and compatible cloud stores, such as Ceph or S3. Spark has the APIs which facilitate programming in R, Python, Java and Scala (Section 5.1).

SparkStreaming is a software component in Spark stack. The software processes real-time streaming data using micro-batches. SparkStreaming processes DStream (or discretized stream) which consists of a series of RDDs as the real-time data.

This chapter focuses on streaming data concepts, stream analytics and real-time analytics.

7.1 ! INTRODUCTION

Data stream in general means continuously flowing data in sequences. A theoretical definition of stream is an unbounded sequence of data items and records, which may or may not be related, or correlated with each other. Examples of the stream are computer network traffic, data of stock quotations, online videos, telephone conversations, transactions in database and transmission of sensors data.

Streaming data originates from several sources. Examples include data and images from satellites, IoT devices, Internet websites and social media posts. Many applications require streaming data for various services. Processing of data stream helps extraction of knowledge-structures from the continuous, rapid flowing data stream.

Some important key terms and their meanings are given below:

Apache® Spark™ refers to a fast and general compute engine. Apache® Spark™ powers analytics applications up to 100 times faster. It supports HDFS compatible data. Spark has a simple and expressive programming model. The multiple languages, Python, and Scala shells provide *great ease in programming for complex analytics, machine learning, and other solutions*.

Spark software stack refers to SparkSQL, SparkStreaming, Spark Arrow, SparkR, MLlib, and GraphX.

Scalability can refer to many different system parameters, such as how much added traffic can it handle, how easy is it to add more storage capacity, or even how many more transactions process.

This chapter describes methods of processing, analyzing, mining the streaming datasets and SparkStreaming—a real-time analytics platform for Big Data. Section 7.2 describes the concept, model, architecture, management of data stream and gives examples of sources. Section 7.3 describes stream computing aspects—sampling, filtering and counting distinct elements in a stream. Section 7.4 describes the methods of frequent itemset stream analytics, handling large datasets and association rule mining. Section 7.5 describes real-time analytics platform—Apache SparkStreaming, and case studies on real-time sentiment analytics and stock prices analytics.

7.2 | DATA STREAM CONCEPTS AND DATA STREAM MANAGEMENT

The following subsections describe the concept, model, architecture and management of data streams.

LO 7.1

Data stream concept,
modeling, architecture,
management, sources,
stream querying and issues

7.2.1 Data Stream Concepts

A stream is a sequence of data elements or symbols made available over time. Data stream transmits from a source and receives at the processing end in a network.

An application processes a data stream differently. Processing is in micro-batches instead of processing batches. Processing of stream can be comprehended as filling milk in bottles on a conveyor belt and capping them, one at a time successfully rather than in a large batch at the same time.

Standard operations do not apply on stream as they may have unlimited or infinite data, and at an instance may not be available completely. Formally, streams are partial data (potentially unlimited), and not finite data. Programmers use the term “stream” in computer science in several ways. Some of these are:

- (i) Stream is communication of bytes or characters over sockets in a computer network.
- (ii) A program uses stream as an underlying data type in inter-process communication channels.
- (iii) Stream Java class objects perform I/Os. Java encapsulates Stream Classes in java.io package. Java defines two types of streams: Byte Stream and Character Stream. Java has several IO classes for inputs and outputs, such as ObjectOutputStream, ObjectInputStream, and FilterOutputStream. A stream object is a logical interface to a file. That file can refer to a disk file, screen, keyboard, port, or a file on a secondary storage device.
- (iv) A stream is a source or sink of data in UNIX. It usually comprises individual bytes or characters. Stream behaves as an abstraction when reading or writing files, or communicating over network sockets. UNIX recognizes three standard input and output streams called standard input (stdin), standard output (stdout) and standard error (stderr).
- (v) A UNIX utility sed (stream editor) parses and transforms text using a simple compact programming language.
- (vi) A Linux stream is a data transfer entity in a Linux shell that transfers data from one process to another through a pipe, or from one file to another as a redirect.
- (vii) Perl and Python implement the stream as an iterator. An iterator is a useful abstraction of an input stream.
- (viii) A stream is an abstraction of a construct that sends or receives an unknown number of bytes in some programming languages. C++ uses this abstraction to perform input and output. Here, a stream is an entity where a program can either insert or extract characters.
- (ix) Oracle streams enable information sharing. A message is a unit of shared piece of information. Oracle streams can share the messages in the stream. The stream propagates information within a database or from one database to another.

- (x) HTTP Live Streaming (also known as HLS) is an HTTP-based media streaming communication protocol implemented by Apple Inc. It is a part of Apple's QuickTime, Safari, OS X and iOS software.

Streaming is a term which is popular in the field of media as well. Here, streaming means listening to music or watching video in 'real time' instead of first downloading and then listening or viewing the content. The contents transmit in a compressed format over the Internet and the receiver plays the contents instantly. A continuous stream of data flows between the source and receiver ends, and is processed in real time.

7.2.2 Data Stream Model

Stream is data in motion. Three approaches for updating the end-points (sinks) are (i) non-overlapping, (ii) slow (batch processing) and (iii) fast (near real-time). Following are the different ways of modeling data stream, querying, processing and management.

(a) Graph model Stream can be modeled as a graph of nodes connected by the edges. The edges model the stream of data moving between the operators. The data processes at the node. Each node in the graph is an operator or adapter. The node can have inputs, zero inputs and outputs or zero outputs. One node output connects to the input of the next node. Figure 7.1 shows graph-based stream as data model for processing at an operator or adapter.



Figure 7.1 Graph-based stream data model for processing at an operator or adapter

(b) Relation-oriented stream-tuples model Stream dataflow can be modeled as tuples flow. Individual data items may be relational tuples in a data stream model.

Data sink may be Parquet nested column-oriented data stream or RDBMS row-oriented storage in tables. *Parquet* is nested hierarchical columnar-storage concept. Nesting sequence is table, row groups within a table, column chunk within the row groups, page chunks with the column chunk. Chunk pages are inside a column chunk, column chunks inside a row group and row groups inside a table (Section 3.3.3.5).

Examples of tuple model stream of data are Borealis, STREAM and TelegraphCQ. Figure 7.2 shows relational tuple-based data stream model. Arrow shows the stream S flowing from the source to sink. S consists of infinite (means unbounded) time-ordered multiple sets of tuples. Each tuple consists of (Timestamp t , Key, Values). Sink is Parquet nested column-oriented or RDBMS tabular store.

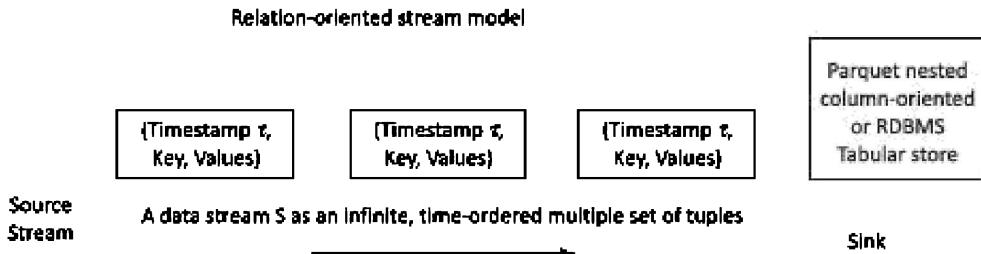


Figure 7.2 Relation-oriented stream-tuples model (Time stamp for real-time streaming data)

Stream dataflow can be modeled as Parquet nested sequences (Section 3.3.3.5). The sequences in data stream are page chunks nested in column chunks in each column. First the (i) page chunks, which nest in a column chunk, then (ii) page chunks of the next column chunk, then (iii) the remaining column chunks which nest at a row group in the table, then (iv) the next row group, and so on. Thus, all Parquet tabular data transfer as the stream in the column-oriented data stream.

Processing in data stream model can also include some data from conventional stored relations, if required. That means queries on data stream may perform join operations between data stream and stored relational data.

A data stream is a potentially unbounded and time-ordered sequence of data items (relational tuples) in the data stream model. The receiving software receives the sequences in order and sees the data items only once. Each tuple

consists of a set of attributes, like a row in a database table. The tuples have a schema-like traditional database. One of the attributes in the tuple schema is a timestamp, usually represented by t . The timestamp denotes the logical arrival time of the tuple into the system.

(c) Object-based data stream model Stream dataflows can be modeled as objects. Cougar and Tribeca are two examples of object-based data stream models. Cougar models sensors' data as a stream of objects. Tribeca models the network monitoring data as a stream of objects.

(d) XML-based data stream model NiagaraCQ is an XML-based data stream model. It provides scalable continuous query processing over XML documents. It performs operations over millions of simultaneous queries by dynamically grouping them according to their structural similarities.

(e) Window-based data stream model Stream data direction can be towards fixed window, sliding window or landmark window-sinks (end-points) [Window means a time window during which the data stream is looked at an instance. Suppose an application software queries streaming data at each successive 2^{20} time-units in time = T_{window} . Then, 2^{30} bits queried for 1 or 0 takes ($2^{10} = 1024$) $\times T_{\text{window}}$ time units (minimum). One time unit corresponds to a time interval in which a bit 1 or 0 can arrive in the stream. Application query caches the arriving bit in one time unit. Stream data unpredictably changes in both size and frequency, and thus it can happen in certain time units that no bits may be arriving. [$2^{10} T_{\text{window}}$ time-units may receive less or much less than 2^{30} bits.]

Examples of stream processing modeling systems Examples of modeling systems for stream processing are:

1. COUGAR is a sensor DBMS. It models the sensors' data as abstract data type (ADT) and sensors data as time series.
2. Tribeca is an older stream processing system for network monitoring applications. An application program expresses queries using a specific dataflow-oriented query language. It cannot support join operations. Tribeca provides windowed aggregates. It also supports other operators that split and merge streams. Group-by splits the streams whereby union merges the streams.

3. *Borealis* is a distributed stream processing engine from Brandeis University, Brown University and MIT. *Borealis* builds on their previous engines, *Aurora* and *Medusa* for stream processing.

Borealis, current version includes various modules, such as a stream processing engine. The version provides the core functionality of low latency in stream processing using a rich set of stream-oriented operators. *Borealis* includes coordinator, load manager, load shedder and fault tolerance modules. Also includes graphical query editor, system visualizer, and stream connection generator.

4. *STREAM* is a data stream management system developed at Stanford University. The project reinvestigated data management and query processing in the presence of multiple, continuous, rapid, time-varying data stream. The *STREAM* prototype supports continuous queries over stream as well as stored relations. To achieve this, *STREAM* supports three types of operators: stream-to-relation, relation-to-stream and relation-to-relation.
5. *TelegraphCQ* project developed at University of California, Berkeley, is a suite of technologies for continuously adaptive query processing. *TelegraphCQ* handles a large number of queries over high volume, highly variable data stream. It continuously processes the incoming data without any storage.

7.2.3 Architecture

Big Data stores at the:

- (i) HDFS (Distributed at data nodes in clusters), or
- (ii) DFS compatible data store, such as HBase, Cassandra, Ceph or S3 (Section 5.2).

Figure 7.3 shows a query processing architecture.

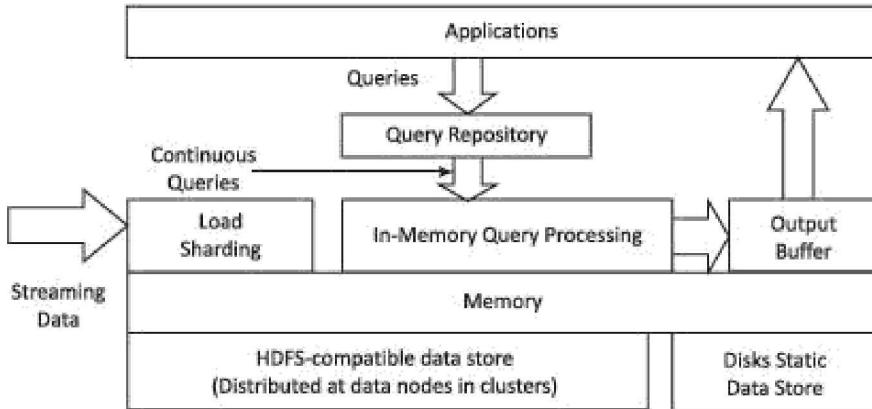


Figure 7.3 Data stream architecture for processing

Figure 7.3 shows that large data blocks in received stream store at HDFS compatible data store or static data at disk. Data shards load at memory from data store or disk for future uses (Section 3.2). The figure also shows that streaming data shards load at memory in real-time applications. A user application uses a query repository, which continuously sends queries for processing of the shards in-memory. The responses of queries save at an output buffer before they are finally retrieved by the application.

The processing model can also use hybrid architecture, referred as *lambda architecture* for processing streaming data and back-end jobs at the same time. The system manages stream flow over real-time data until data elements are pushed to a batch system. The data then become available to access and process as batch data from disk static data store or HDFS.

7.2.4 Data Stream Management System (DSMS)

Fundamentally basis of traditional data management systems are on the notion of determined and static data storage. The streaming data basis is altogether different. The data requires collection and parsing before using and deletion from the system.

Responses against the queries are precise in static DBMS. Streaming data can unpredictably change in both size and frequency, and thus results into approximate responses.

Management and processing of streaming data are different from traditional DBMSs. These systems build primarily on the concept of persistence and static

data collections while the streaming data require parsing and processing at once when they arrive in the system.

Data Stream Management System (DSMS) is a data-intensive application in which the data models as a transient data stream. DSMS is an application program that manages streaming data. The usage of the program is different than using of persistent relations in DBMS.

Some popular applications of DSMS are network monitoring, telecommunications data management, web applications and sensor networks. Individual data items model as relational tuples in the data stream model. For example, data of network measurements, call records, web page visits, and sensor readings. Table 7.1 highlights the differences between DBMS and DSMS.

Table 7.1 Comparison between DBMS and DSMS

DBMS	DSMS
Stores sets of records with no pre-defined time concept	Provides on-line analysis of rapidly changing stream of data
Suitable for applications that require persistent data storage and complex querying	Suitable for real-time, continuous, ordered (arrival time or timestamp) sequence of data elements. Also, for data that is large to store entirely and continuous querying environment
Persistent relations (relatively static, stored)	Transient stream (on-line analysis)
One-time queries	Continuous queries
Requires random access	Implements sequential access
Unbounded disk storage	Bounded main memory
Only current state is important	Past or historical data is important
No real-time services requirements	Real-time requirements
Relatively low update rate	Very high arrival rate (usually in multi-GBs)
Assume precise data	Data get stale or imprecise later
Access plan determined by query processor, physical database design	Unpredictable data arrival and varying characteristics of the data

Traditional DBMSs do not directly support real-time continuous queries. The data is available on disk or memory for random access to them. Some or all input data that are to be operated on and arrive as one or more continuous data stream in the data stream model.

Data stream differs from the conventional stored relational model in the following ways:

- The data elements in the stream arrive in real time.
- Size of the data stream is unbounded.
- The data processes in the order in which the data elements arrive.
- The processed elements delete or archive.
- The buffer is relatively small than the size of the data stream and not accessible easily.

These are two types of data streams:

1. **Transactional data stream:** They carry data related to the interactions between different entities. For example:
 - (i) ATM transactions – Withdrawals/deposits by users from bank accounts
 - (ii) Telecommunication – Phone calls by callers to dialed numbers
 - (iii) Web access by clients of resources at servers.
2. **Measurement data stream:** They carry data related to measured values or metric of entity states. For example:
 - (i) Sensor networks – Traffic density values, presence, or absence of obstacles in the path
 - (ii) IP network – IP packet in and out at router interfaces
 - (iii) Climate data – Temperature and moisture records.

Many systems support handling of streams. Several major research projects relate to relation-oriented stream databases.

Following are examples of commercial databases for streams:

- (i) **StreamBase** (commercial version of Aurora/Borealis)

- (ii) **Truviso** (commercial version of TelegraphCQ); Cisco acquired it in May 2012 extending existing relational databases to build TelegraphCQ and Truviso syntaxes. They are extensions of PostgreSQL that incorporate data stream
- (iii) **TIBCO Business Events**, Oracle Business Activity Monitoring

7.2.5 Example of Sources of Streams

The sources of streaming data range from simple computer programs to Internet of Things (IoT) applications. The sources of stream include sensory machines, satellites, instruments, IoT application areas, websites, published information from service providers and social media posts.

Some useful applications of data stream are:

1. Making data-driven marketing decisions in real time. It requires the use of data from trends analyses of real-time sales, and analysis of social media, and the sales distribution.
2. Monitoring and detection of potential failures of system using network management tool
3. Monitoring of industrial or manufacturing machinery in real time
4. A sensor network or IoT controlled by another entity, or a set of entities
5. Watching online video lectures, and rewinding or forwarding them
6. Subscribing to the daily news alerts, weather forecasting services or other similar subscription based services
7. Using location based services, such as finding nearest point-of-interest (POI)
8. Getting location-based advertisements or notifications
9. Watching on-demand movie, listening to online music, watching television
10. Navigation using GPS
11. Playing online games
12. Response stream from a web server

13. Using social networks, such as Facebook and Twitter
14. Traffic management, pollution control, flight traffic control, war field surveillance using sensor networks.

DSMS deals with streams and processes them differently from traditional data management systems. A traditional system builds primarily on the concept of persistence and static data collections. Streaming data requires traversing and processing at once before collection or deletion in the system.

7.2.6 Stream Queries

Data is static in a relational database. Thus, applications send queries to the database and obtain the results. They are one-time queries or transient queries. Data in stream changes frequently. The results of the queries against the stream also change. The queries are defined as continuous queries or persistent queries. They process continuously as data continue to arrive.

Their query processing results can be obtained in two forms. The results store and update when data arrives or they make data stream for the results themselves. For example, aggregation queries mostly use the first form, while join queries may use the stream form.

The queries may also be classified as predefined and ad hoc queries. Continuous queries are generally predefined and therefore register with the database server.

Ad hoc queries can be issued online along with the flowing data stream. They can be either one-time queries or continuous queries. Ad hoc queries make the design of a DSMS difficult. Firstly, they do not optimize since they're not known in advance. Secondly, they may require reference to an already streamed (or discarded) data for correct results.

Another issue related to queries in DBMS and DSMS is that the former mostly leads to exact query results while the later mostly approximate-query results.

Query Languages

1. **Relation-based** query language is based on SQL-like syntax. These languages provide better support for windows and ordering. Examples are Esper, CQL (STREAM), StreaQuel (TelegraphCQ), AQuery and GigaScope.

2. **Object-based** query languages are based on object-oriented stream modeling. These languages classify stream-elements according to type hierarchy. Examples are Tribeca and COUGAR.
3. **Procedure-based languages** are those where user functions (procedures) specify the dataflow. For example, Aurora provides graphical interface to users for constructing query plans.

Examples of query languages are given below:

STREAM Continuous Query Language (CQL) CQL developed at Stanford is an extension of SQL. The following example gives usages of CQL syntaxes.

EXAMPLE 7.1

Consider a network security monitoring system. (i) How does STREAM CQL create a stream? (ii) How does STREAM CQL remove a stream? (iii) How does STREAM CQL use time within the query?

SOLUTION

(i) Creating stream: CQL syntax is:

```
CREATE STREAM network_admin(  
    communication_time TIMESTAMP,  
    ticker_symbol VARCHAR (10),  
    num_packets INTEGER,  
    bytes_per_packet NUMERIC (9, 0)  
) ;
```

(ii) Removing stream: CQL Syntax is:

```
DROP STREAM network_admin;
```

(iii) Actual data stream generally arrives over network and must be in a specific format for the database to use it.

STREAM CQL stream uses time within the query:

```
SELECT ticker_symbol,
```

```

        SUM(num_packets      *      bytes_per_packet)      /
        SUM(num_packets)
        FROM network_admin [RANGE 5 MINUTES]
        GROUPBY ticker_symbol;
    
```

The following example gives usages of Truviso syntaxes.

EXAMPLE 7.2

How does Truviso syntax create stream and use system-generated timestamps?

SOLUTION

- (i) Create a stream with time stamp:

```

CREATE STREAM network_admin(
    communication_time      TIMESTAMP      CQTIME      USER
GENERATED,
    ticker_symbol VARCHAR (10),
    num_packets INTEGER,
    bytes_per_packet NUMERIC (7, 2)
);
    
```

- (ii) Queries can be issued against both relations and stream. If a stream is involved, then it specifies the window as follows:

```

SELECT ticker_symbol,
        SUM(num_packets      *      bytes_per_packet)      /
        SUM(num_packets)
        FROM network_admin< VISIBLE '5 MINUTES' >
        GROUPBY ticker_symbol;
    
```

Example 7.3 explains usages of TelegraphCQ TIMESTAMP COLUMN modifier.

EXAMPLE 7.3

How is query in TelegraphCQ used by giving a separate window specification?

SOLUTION

TelegraphCQ has a separate window specification:

```
SELECT ticker_symbol,  
       SUM(num_packets      *          bytes_per_packet) /  
           SUM(num_packets)  
  FROM network_admin  
 GROUPBY   ticker_symbol    WINDOW    network_admin    [ '5  
 MINUTES' ];
```

7.2.7 Stream Processing Issues

Stream processing refers to a method of continuous computation that takes place as data is flowing through the system. The processing can be helpful to gather statistics and/or monitor the streaming data. It can also, help in studying the source of streaming data and forecast the future behavior.

The unbounded size, frequency, velocity and variety of data in stream add challenges to the system. Following issues surface during stream processing:

Size of the streaming data not fixed Batch or static processing is a method of stored data processing. The calculations of execution time and usage of memory for algorithm are easy because processing of static data depends on the size. The size of streaming data size is variable and is rather unbounded. Therefore, algorithm works differently. Algorithms cannot iterate over the complete data. The processing can be possible on a piece of data comprising specific data elements or recent data elements.

Need of scalable processing Very high data volume requires scalable processing. As applications grow, their data processing needs also grow. Fulfilling demands of growing users can accelerate data processing requirements as the application demand grows.

Size is a vital aspect of scale that needs consideration in case of processing large datasets and distribution systems. System capacity need to increase for handling greater amounts of data, commonly referred as system scalability issue.

Variation in the frequency of data stream The frequency of data stream varies significantly over time. These variations are unpredictable or depend on social and public trends. For example, streaming data on social networks such as Facebook and Twitter can increase in volume during holidays or festivals. The variations can be periodic also, for example, in the evenings or weekends.

Need of near real-time processing The streaming data management systems require near real-time processing. Managing and processing data in motion is a typical capability of streaming data systems. The data requires processing when the data is in motion on place after the collection. The analogy can be found in sensor data processing. The computations need completion in real time. The processing needs to be fast for streaming sensor data.

Need of processing large data streams from different domains Data streams can vary with domain. The streams can source from several applications. Data arrives as large data streams from different domains. For example, satellite data, scientific instruments, sensors, social networks, stock data, process industries, traffic controls and network logs. Processing of variety (formats, types, compression) of data is another challenge.

Need of events-processing The processing may be required for different events representing measurements, such as position, particle mass, temperature, number of tweets per minute, stock update, number of items manufactured during last one hour, number of vehicles passed in an hour, number of login on Amazon on a festival day, etc.

Need of filtering to eliminate undesirable data elements Another important processing need is for searching the required data from a data stream and filtering the stream to eliminate undesirable data elements.

A restriction in data streaming systems is must. They should carry out relatively simple computations, say one record processing at an instance. The other requirements may be near real-time computations, at times in-memory and independent computations.

The processing function often provides the service to a system or a stream source. The function does not interact with system or source (one-way

communication). They do not even provide any feedback of the system. Most stream-based systems are also subscriber-based systems.

7.2.8 Real-time Processing, Stream Processing and Batch Processing

A system is a real-time system if it provides the output guaranteed within hard “real-world” time deadlines. Software, which continuously processes a stream or stored stream, achieves nearly real-time performance. Time limitation is not a concern in stream processing. No fixed time deadline exists on the output of the system after receiving an input. These requirements for streaming data processing are reasonably different from batch processing. Data management and analysis are inclusive in batch processing.

7.2.9 Summarizing Streaming Processing Needs

Stream processing needs are:

- (i) Computations are a function of a single data element or a smaller piece of recent data. They have no access to the complete data.
- (ii) Processing algorithms must be relatively fast and simple
- (iii) Need to complete each computation in near real-time while static processing has more latency
- (iv) Computations are generally independent
- (v) Asynchronous processing, i.e., the source of data does not interact with the stream processing system directly
- (vi) Requirements of high-volume processing with low latency because no information exists about when the next data will arrives.

Self-Assessment Exercise linked to LO 7.1

1. List ten examples from two different domains for data streams.
2. How does a data-stream model as a relation-oriented stream tuples? What are the benefits of the relational tuples model?
3. Draw the data-stream processing architecture.

4. List the characteristics of Data Stream Management System (DSMS).
5. List the features of different types of data stream query languages.
6. What are the issues in data stream processing in case of data with high velocity and volume?

7.3 ! STREAM COMPUTING ASPECTS

LO 7.2

Many applications require Big Data stream computing. Stream computing is a way to analyze and process Big Data in real time to gain current insights. Following subsections describe the methods of stream computing, sampling data in a stream, filtering of data in a stream and counting of distinct elements:

Stream computing aspects—sampling, filtering and counting distinct elements in a stream

7.3.1 Stream Computing

Stream computing has many uses, such as financial sectors for business intelligence, risk management, marketing management, etc. Stream computing is also used in search engines and social network analysis.

The computing pulls the data from the stream, process the data, and streams it back out as a single flow. Such computing is required to process a huge amount of data at a high speed.

Usually, a Big Data stream computing implements in a distributed clustered environment, as the amount of data is enormous. Rate of receiving data in stream is high, and the results are required in real time to take appropriate decisions or to predict new trends in the immediate future.

Stream computing uses algorithms that analyze data in real time at high speed and with accuracy. Stream computing is one effective way to support Big Data by providing extremely low-latency velocities with massively parallel processing architectures.

Stream computing is becoming the fastest and most efficient way to obtain useful knowledge from Big Data. Organizations can react quickly when problems appear, or to predict new trends for the future.

The efficiency of data stream algorithms is measured using some fundamental characteristics:

1. Number of passes (scans) the algorithm must make over the stream
2. Available memory
3. Running time of the algorithm.

Data stream algorithms usually have limited memory availability. They may take certain action until the dataset arrives completely (for which the application is waiting). On the other hand, the usual online algorithms require action as soon as every piece of data arrives.

7.3.2 Sampling Data in a Stream

Sampling in data stream means the process of selecting a few data items from the incoming stream of data items for analysis. Methods of obtaining representative sample data items from a stream can be classified in two categories:

Obtaining a Representative Sample

Two categories of sampling methods are probabilistic and non-probabilistic. First category, probabilistic sampling is a statistical technique used for making a choice of data items for processing. The basis of the choice is the probability of sampling the data items. For example, if probability value chosen is 0.01, the method takes up 1 out of 100 data items for analysis.

Probabilistic sampling technique obtains the representative sample. The sample chosen is an actual representative of the population.

Five probabilistic sampling methods are simple random sampling, systematic sampling, cluster sampling, stratified sampling and multistage sampling.

Second category is non-probabilistic. Non-probability sampling uses arbitrary or purposive sample selection instead of sampling based on a randomized selection. This introduces bias and increases variance to the measurement data.

Reservoir sampling is a random sampling method. The method chooses a sample of limited data items from a list containing a very large number of items randomly. The list is larger than one that upholds in the main memory. An example of reservoir sampling method is given below:

EXAMPLE 7.4

Using the reservoir sampling method, illustrate the probability that the incoming item is stored in the main memory, while choosing a sample of limited data items from a very large number of items?

SOLUTION

Let k be the number of items selected from an infinite stream of data items. Suppose when processing a sequence of items, the program processes one at a time. Hence, for n items, the probability that a new item is in the main memory will be k/n . The algorithm works as:

Select the first k items in memory.

```
for (i > k) {
```

On the arrival of i^{th} item, select a new item with probability (k/i)

Remove an old item at random, each with chance $1/k$ with probability $(1 - k/i)$, retain the old items instead of new item}

Thus,

Items $\leq k$, the probability of item in memory is 1

Item = $k+1$ the probability will be $k / (k+1)$

Item = $k+2$ the probability will be $k / (k+2)$

Concise sampling and **counting sampling** are other uniform random sampling methods. Concise method is like the reservoir sampling method, with a difference that a value that appears once is stored as a singleton, whereas a value that appears more than once is stored as a (value, count) pair. This overcomes the restriction of sample size due to the size of the main memory. The method inserts a new data item in the sample with a probability of $1/n$. If the new item is already present in sample, the count is incremented.

Counting sampling method is the refinement of concise sampling in terms of accuracy. The method maintains the sample in the case of deletion of data items as well. The method reverses the increment procedure by decrementing the count value upon deleting a value. The process may lead to converting back to

singleton when a single value remains in the sample or even removing a singleton if that single value is removed.

General Sampling Problem

Some problems encountered while trying to find a sample of a data item from an infinite length data stream are:

- (i) Unknown size of data set
- (ii) Applications that need continuous analysis, such as surveillance analysis
- (iii) Irregular data rates as in the case of data network analysis.

Varying the Sample Size

Sample size means the number of data items, with respect to the reference number of data items, chosen to make an inference or decision. A large sample size can generally give inference with greater accuracy. However, it may not be feasible to choose a large size.

Procedures for calculating sample sizes are (i) estimation, called confidence interval approach, and (ii) hypothesis testing. Statistics prescribes Chi-squared, T-test, Z-test, F-test, P value for testing the significance of a statistical inference.

The number of tables is available for sample size estimation for making inferences. Estimation is done for the minimum sample size required for accuracy in estimating the key proportions P of data items D for selection.

Some steps taken into consideration are: (i) a reasonable estimate of P to be measured in the study, (ii) the degree of accuracy D that is desired in the study, ~1% -5% or 0.01–0.05, and (iii) the confidence level Z needed from the inference, 95%.

7.3.3 Filtering of Stream

Filtering application identifies the sequence patterns in a stream. Stream filtering is the process of selection or matching instances of a desired pattern in a continuous stream of data.

For example, assume that a data stream consists of tuples. Following are the filtering steps: (i) Accept the tuples that meet a criterion in the stream, (ii) Pass the accepted tuples to another process as a stream and (iii) discard remaining tuples.

Several filtering techniques exist: Bloom Filter and its variants, Streaming Quotient Filter (SQF), Particle filter, Kalman filter, XML filters (such as XFilter, YFilter).

Conditional matching is simple to implement, even in case of streaming data when a tuple is matched with some desired value. For example, if the ‘stock price’ field value is greater than the maximum price in 52 weeks’ in the stream of data, then that price filters out using a single ‘if’ condition. The complexity of computation increases when one has to check duplication of records or match more than one tuple with a loose condition (can be or cannot be accepted). Then, the requirement is revisiting the record that becomes a memory-demanding computation.

The following subsections provide an overview of Bloom filter, one of the popular variants of Bloom filter (Counting Bloom Filter):

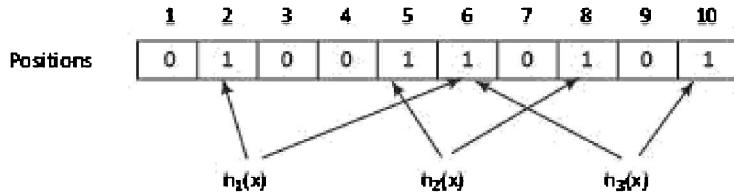
7.3.3.1 Filtering of Stream: The Bloom Filter Analysis

Bloom filter is a simple space-efficient data structure introduced by Burton Howard Bloom in 1970.¹ The filter matches the membership of an element in a dataset. The data structure is a probabilistic representation of a vector that supports membership queries. [Anand Rajaraman *et al.*²]

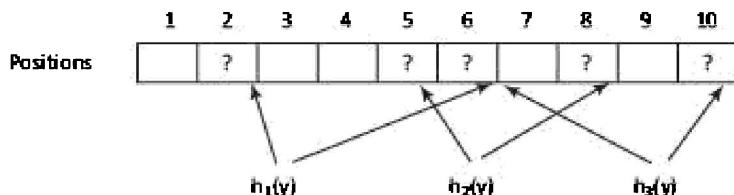
The filter, since then has been widely used in applications, such as database applications, intrusion detection systems, and query filtering and routing applications.

The filter is basically a bit vector of length m that represent a set $S = \{x_1, x_2, \dots, x_n\}$ of n elements. Initially all bits are set to 0. Then, define k independent hash functions, h_1, h_2, \dots, h_k . Each of which maps (hashes) some element x in set S to one of the m array positions with a uniform random distribution. Number k is constant, and much smaller than m . That is, for each element $x \in S$, the bits $h_i(x)$ are set to 1 for $1 \leq i \leq k$. [\in is symbol in set theory for ‘contained in’.]

Figure 7.4 shows an example of a Bloom filter with $k = 3$. The filter is a bit vector of length 10. When a value x is inserted into the Bloom filter, the bits at position $h_1(x)$, $h_2(x)$ and $h_3(x)$ are set to 1. To detect whether a value y is in the filter or not, the bits of position $h_1(y)$, $h_2(y)$ and $h_3(y)$ are thus checked.



(a) Inserting an element x in bit vector (filter) of length $m = 10$



(b) Finding element y is in the filter

Figure 7.4 (a) Inserting an element x in bit vector (filter) of length $m = 10$,
(b) finding an element y in an example of Bloom Filter

The operation results in setting a particular bit to 1 many of times, but only the first operation has an effect.

To check if an item y is in S , the bits at positions $h_1(y), h_2(y), \dots$, and $h_k(y)$ are checked (Figure 7.4 (b)). If any of the bits is 0, undoubtedly y is not a member of S . If all $h_i(y)$ are found to be 1, y may be in S . The bits may have by chance been set to 1 during the insertion of other elements. Thus, chance of incorrect assumption is present with some probability. This is a false positive, where the Bloom filter suggests that an element y is in S even though it is not.

It has been given in the literature that the probability of a false positive is equal to $(1 - e^{-kn/m})^k$.

7.3.3.2 Counting Bloom Filter: A Variant of Bloom Filter

Process of deleting a particular element in the Bloom filter requires that the corresponding positions computed by k hash functions in the bit vector, be set to zero. This may possibly concern other elements stored in the filter, which hash to any of these positions. For example, the bit position 6 is set to one by two hash functions $h_1(x)$ and $h_3(x)$ in Figure 7.4(a). Thus, it is not possible to delete an element stored in the filter.

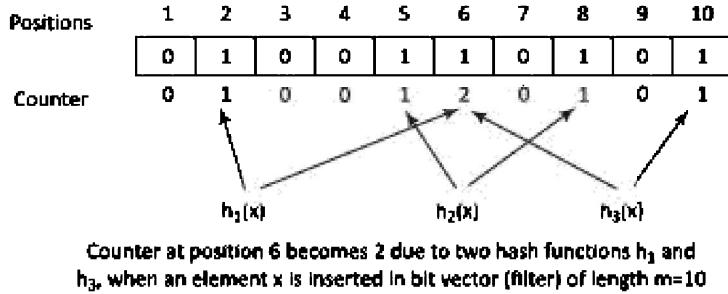


Figure 7.5 An example of Counting Bloom Filter

In order to perform deletion of the element, a method counting Bloom filter, a variant of Bloom filter be used. The counting filter maintains a counter for each bit in the Bloom filter. The counters corresponding to the k hash values are incremented or decremented, whenever an element in the filter is added or deleted, respectively. As soon as a counter changes from 0 to 1, the corresponding bit in the bit vector is set to 1. When a counter changes from 1 to 0, the corresponding bit in the bit vector is set to 0. The counter basically maintains the number of elements that hashed to the corresponding bit by any of the k hash functions.

7.3.4 Counting Distinct Elements in a Stream

The count-distinct problem relates to finding the number of dissimilar elements in a data stream. The stream of data contains repeated elements. This is a well-known problem in networking and databases. Several applications require finding the dissimilar or distinct elements. For example, packets passing through a router, unique visitors to a web site, recurring patterns in a DNA sequence, records in a large database, or elements of sensor networks.

7.3.4.1 Counting Distinct Elements in a Stream and Count Distinct Problem

If n possible elements a_1, a_2, \dots , and a_n are present then for an exact result n spaces are required. In the worst case, all n elements can be present. Let m be the number of distinct elements. The objective is to find an estimate of m using only s storage units, where $s \ll m$.

The example below gives an algorithm to compute distinct elements.

Assume data items set (A, B, B, C, C, D, B, A, A, D, C, B, C). Assume m is the number of distinct elements. (i) What are the distinct elements? (ii) Write an algorithm to compute m_t .

SOLUTION

- (i) Number of distinct elements, $m = |\{A, B, C, D\}| = 4$
- (ii) Assume D will contain the list of distinct elements, a_i where $i = 0, 1, 2, \dots, m$. Algorithm is as follows:

Initialize a Counter $c = 0$;

Initialize a data structure say, D in which insertion of an element can be done easily (for example, hash table or search tree);

Repeat for each element a_i of stream

{ If a_i is not in D

 add a_i to D;

 increment c;

end if; }

until stream is over;

output $m = c$;

If m is not very large, D fits in the main memory and an exact answer can be retrieved as shown in Example 7.5. However, the approach does not scale for bounded storage, or if the computation performed for each element a_i is forced to be minimized. Another constraint that takes place in streaming data processing is that the algorithm only observes each input element once.

Several proposed streaming algorithms use a fixed number of storage units in such case; for example, bitmap algorithm. The algorithm uses a limited amount of memory for solving the counting distinct element problem.

The following example gives the bitmap algorithm to compute distinct elements.

EXAMPLE 7.6

- (a) Write the bitmap algorithm for computing m , the number of distinct elements.
- (b) Find m in the following data stream: A, B, B, D, A, A, E, B, H, B, A, F, D, E.

SOLUTION

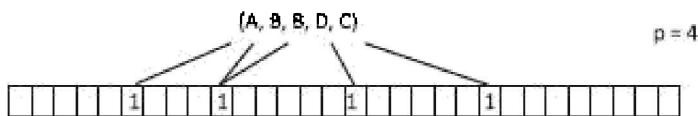
- (a) Assume a large binary array (bitmap) of size S with all members initializing to 0.

Choose a hash function h such that the value $h(i)$ is uniformly distributed on $[S]$:

$$\{1, \dots, n\} \rightarrow \{1, \dots, S\}.$$

Apply the hash function on every element (i) of data stream to compute $h(i)$ and mark that position in the bitmap with a 1.

Count the number of positions in the bitmap with a 1 and call it p .

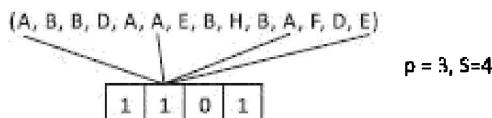


$$\text{Thus, } m = S \cdot \ln \left(\frac{S}{S-p} \right), \dots \quad (7.1)$$

Let $S = 4$ and $h(i) = i \bmod 4$, such that:

$h(A) = 1, h(B) = 2, h(C) = 3, h(D) = 0$, and so on.

Thus the bitmap (drawn for $h(i) = 1$ in the following figure) will be:



- (b) Using equation (7.1), the estimated number of distinct elements = $4 \times \ln(4/(4-3)) = 5.55 \approx 6$

Streaming algorithms have several applications, such as monitoring packet flow in networks, counting the number of distinct elements in a stream,

estimating the distribution of flow sizes and estimating the size of a join in databases.

7.3.4.2 The Flajolet–Martin Algorithm

Flajolet–Martin (FM) algorithm approximates the m , number of distinct (unique) elements, in a stream or a database in one pass. The stream consisting of n elements with m unique elements runs in $O(n)$ time and needs $O(\log(m))$ memory. Thus, the space consumption calculates with the maximum number of possible distinct elements in the stream, which makes it innovative.

Following are the features of the FM algorithm:^{3,4}

- (i) Hash-based algorithm.
- (ii) Needs several repetitions to get a good estimate.
- (iii) The more different elements in the data, the more different hash values are obtained
- (iv) Different hash values suggest the chances of one of these values will be unusual [the unusual property can be that the value ends in many 0s (alternates also exist)].

The following example demonstrates an estimation of number of distinct elements in a stream using the FM algorithm:

EXAMPLE 7.7

Find an estimation of the number of distinct elements in the following stream:

$$s = 2, 3, 1, 2, 3, 4, 3, 1, 2, 3, 1, 4$$

SOLUTION

Consider a hash function, say $f(a) = 7s + 2 \bmod 5$

Now apply hash function on the input stream, perform the bit calculation and trailing zeroes, to get:

Apply hash function on input stream				Binary equivalent	Trailing zeroes
f(2)	$7*2+2 \bmod 5$	$16 \bmod 5$	1	001	0
f(3)	$7*3+2 \bmod 5$	$23 \bmod 5$	3	011	0
f(1)	$7*1+2 \bmod 5$	$9 \bmod 5$	4	100	2
f(2)	$7*2+2 \bmod 5$	$16 \bmod 5$	1	001	0
f(3)	$7*3+2 \bmod 5$	$23 \bmod 5$	3	011	0
f(4)	$7*4+2 \bmod 5$	$30 \bmod 5$	0	000	0
f(3)	$7*3+2 \bmod 5$	$23 \bmod 5$	3	011	0
f(1)	$7*1+2 \bmod 5$	$9 \bmod 5$	4	100	2
f(2)	$7*2+2 \bmod 5$	$16 \bmod 5$	1	001	0
f(3)	$7*3+2 \bmod 5$	$23 \bmod 5$	3	011	0
f(1)	$7*1+2 \bmod 5$	$9 \bmod 5$	4	100	2
f(4)	$7*4+2 \bmod 5$	$30 \bmod 5$	0	000	0

If output is zero, then trailing bits are also zero

- (i) The maximum number of trailing zeros from the binary equivalent trailing zero values, $r = 2$.
- (ii) The distinct value $R = 2^r = 2^2 = 4$
- (iii) Therefore, $R = 4$ means there are four(4) distinct values as 2,3,1,4.

7.3.4.3 Combining Estimates— Space Requirements

Different hash functions result in different estimates of m (the number of distinct elements). Thus, one needs to combine all these estimates. Two ways for combining estimates are described below:

One way is to take the average of the values of R , computed from different hash functions. (Example 7.7) Taking the average can be over estimation, thus cannot be a good solution. Another way is to take the median of all the estimates (taking the median is almost correct, but is always a power of 2).

Therefore, combination of the two ways mentioned above can be a good way to combine the estimates. Thus, build the groups of hash function and take their average, then take the median of the averages.

Processing multiple data stream and combining limits using in-memory processing is feasible.

7.3.5 Estimating Moments

Recall Sections 6.2.4 and 6.2.5. Assume a random variable X where X refers to a variable, such as number of distinct elements x in data stream. Assume that variable x has probabilistic distribution in values around the mean value \bar{x} . Probabilistic distribution means probability of variable having value found = x varying with variable X . Expected value among the distributed X_i values where i varies from 0 to n will depend upon the expected distinct element count. Expected value will be m for expected number of distinct elements in the data stream, and much less than m for wide variance.

The variance is the square of the standard deviation in m from the expected value, the second central moment of a distribution, and the σ^2 or $\text{var}(x)$ represents covariance of the random variable with itself. The method computes variance for formula:

$$\sigma^2 = \sum(x_i - \bar{x})^2/n \quad \dots (7.2)$$

Moments (0, 1, 2 ...) refer to expected values to the powers of (0, 1, 2 ...) of random-variable variance (Section 6.2.5).

Let $P(x_i)$ is probability that $m = m_i$. Sum of probabilities $P(m_i)$ over all possible n values of x is 1. 0th moment is always 1.

7.3.6 Counting of 1's in a Window

Streaming data are fundamentally continuously generated data. Continuous data stream may be infinite. A good example is network traffic analysis. Here, millions of packets arrive per second, and hundreds of concurrent queries are raised per second.

Infinite Stream Processing

Figure 7.3 showed data stream architecture for processing queries. Volume of data is too large that it cannot be stored. Hardly a chance exists to look at all of it. Stream processing is important for applications where new data arrives frequently. Important queries may be likely to ask about the most recent data or summaries of data.

Sliding Time Window Method for Data Stream Processing

The sliding window model for data stream algorithms is a popular model for

infinite data stream processing. (Window refers to time interval during which stream raised and processed the queries). The receiving of data elements is taking place one by one. Statistical computations are over a sliding time-window of size N (not over the whole stream) in time-units. Window covers the most recent data items arrived. Assume that t is the time interval, which a query processing algorithm needs to cache a bit, 1 or 0. Then window time interval T_{window} for raising the queries and processing equals to $N \times t$.

Sliding window focuses on recent data and hence provides more significant and relevant data in real-world applications. The network traffic analysis, requires analysis based on the recent past. This is more informative and useful than analysis based on stale data.

A useful model of stream processing is the one in which queries are processed for a window of length N , where N corresponds to the most-recent elements received. Usually it is so that N is very large and cannot be stored on storage device, or there are so many streams that elements from windows for all cannot be stored.

Counting of 1's Problem

Recall Section 7.2.2 (e). Consider a counting problem. For a given stream of 0's and 1's, "How many 1s are present in the last k bits?" where $k \leq N$. The obvious solution is to store the most recent N bits. When new a bit comes in, discard the first bit. This will result into the exact answer. What happens if one cannot have enough memory to store N bits? For example, when the stream processor is processing, assume N is 1 Billion. Here, the solution can be an approximate answer. An algorithm called Datar-Gionis-Indyk-Motwani (DGIM) algorithm is a solution for such problem in counting.

7.3.6.1 DGIM algorithm

DGIM algorithm suggests that store just the $O[\log_2(\log_2 N)]$ bits per stream. The algorithm uses the concept of time buckets. A time window divides in a number of buckets. It is different from hash buckets.

Assume that N_b = time units of specific duration in which stream 1s and 0s arrive. When N time units elapse, the bucket ends. A bucket stores $O(\log_2 N_b)$ bits, and the count of number of 1s between its beginning and end of the bucket (which is the size of the bucket) [$O(\log_2(\log_2 N_b))$]. A bucket in the DGIM

algorithm is a record, which consists of (i) a timestamp placed at a position at which the rightmost bit (which is the most recent bit) arrives. Timestamp consists of $O(\log_2 N_b)$ bits, [Suppose the number of bits 1s as well as 0s arrived in a given bucket time = $1024 \times 1024 \times 1024 = 2^{30}$, then the timestamp will be 11110 (= 30 decimal), and (ii) count cnt of 1s between beginning and rightmost bit, [Suppose cnt during that time = 2^{24} , then cnt = 11000 (= 24 decimal). Maximum value cnt = 29 when all bits arrived happens to be 1s and none 0s when the bucket time 2^{30} time units.] The algorithm prefixes cnt. The succeeding bucket of $2^{(2 \times \text{cnt})}$ time units can be less than the earlier. Earlier arriving buckets are, therefore, not smaller than the later [Anand Rajaraman *et al.*²].

The size of the bucket restricts to the power of two. Buckets do not overlap in timestamps. Buckets are sorted by size (number of 1's). The algorithm uses either one or two buckets with the same power of 2 number of 1's. The error is just equal to the time taken in storing the bucket timestamps and cnt.

Suppose the last bucket has size $2^{(2 \times \text{cnt})}$, then $(2^{\text{cnt}} - 1)$ of its 1's are still within the window, an error of at most that much can be there. Minimum one bucket is of size not less than $2^{(2 \times \text{cnt})}$. Thus, the error can be at most 50%.

Further extensions to the algorithm are also found in literature suggesting the use of the algorithm discussed in this section to handle aggregations more general than counting 1's in a binary stream. The next subsection discusses another important data stream algorithm using the sliding window model.

7.3.7 Decaying Windows

Anand Rajaraman *et al.*² describe the details of the decaying windows method. Decaying windows are useful in applications which need identification of most common elements. The use of the decaying window concept is when more weight assigns to recent elements.

The technique computes a smooth aggregation of all the 1's ever seen in the stream, with decaying weights. When it further appears in the stream, less weight is given.

The effect of exponentially decaying weights is to spread out the weights of the stream elements as far back in time as the stream flows.

Self-Assessment Exercise linked to LO 7.2

1. Why does processing of data stream require sampled datasets in place of all datasets? How is a representative sample taken from a stream?
2. What are the uses of mean, variance, moments, probability distribution and standard deviation in stream computing?
3. What is the information obtained after filtering of data stream?
4. How is the sliding window used for data stream processing?
5. Why is decaying window with decaying weights used for aggregation of all the 1's ever seen in a stream?

7.4 | FREQUENT ITEMSETS

Frequent itemsets and frequent patterns have several uses. The following subsections describe the methods of finding frequent itemsets and handling of larger datasets in the main memory, and algorithms to count the instances of frequent itemsets in streams containing those sets:

LO 7.3

Methods of frequent itemsets analytics and handling large datasets, association rule mining for finding and counting instances of presence of those itemsets

7.4.1 Finding Frequent Itemsets

The computational model of finding frequent itemsets typically consists of mining the number of itemsets in a flat file system. Assume that a Department of Computer Science is offering five courses and students have different computer subjects. Figure 7.6 shows the organization of the courses.

PG Computer Science Students Basket 1	PG Computer Applications Students Basket 2	UG Computer Science Students Basket 3	UG Computer Applications Students Basket 4	UG Information Technology Students Basket 5
Python	Python	Python		Java
Java	Numerical Analysis	Databases	Python	Data Communication
Big Data Analytics	Java	Java	Big Data Analytics	Databases

Figure 7.6 Courses organization

Let us define that if an itemset (subjects-set) is present in at least 3 out of 5 course baskets, then that subjects set is a frequent itemset [Use that criterion for finding whether Java and Python subjects set is the frequent itemset]. If the support threshold, S_{th} is raised to 0.8 (80%), then that subject set does not qualify as a frequent itemset. It is not necessary that an itemset is present in a basket once only, but its presence is counted as once only though the itemset may be present 2, 3 or more times.

Let us consider an association rule. Let us assume SS is a subject set. Let a subject is subj. Consider the association rule that $SS \rightarrow \text{subj}$. It implies that if SS is present in a basket, then subj is likely to appear in that basket.

Frequent sets basket counting problem has many applications: If a sports shop is selling badminton racquets, then it is likely to sell shuttlecocks as well. A car company sells five models through thousands of car showrooms. It wants to analyze in what areas, the Jaguar as well as Zest cars are selling frequently, considering $S_{th} = 80\%$.

Assume (Java, Python) is a subject set SS. The students in a computer course if taught Java subject for study, they may be taught Python as well. Association rules are $(\text{Java}, \text{Python}) \rightarrow \text{Java}$ and $(\text{Java}, \text{Python}) \rightarrow \text{Python}$. Association rule mining means finding the course baskets in which both the association rules apply. Figure 7.6 shows 3 course baskets where both the pairs of rules apply. How does number 3 find? Assume m is number of course baskets and nB is number of baskets where that subject set taught. Following can be the association-rule mining method:

Initialize $nB = 0$;

For each i from 1 to m, [if (Java is a subject in a

```
course-basket CB (i)) then if (Python is also a subject  
in CB (i)) then nB = nB + 1;]  
if (nB >= Sth) then subject-set SS is frequent itemset.
```

The present example considers the number of associations, p just two. What about a frequent pattern in which p is ten or more. Remember that the number of items can be 12000K (Products at Amazon) or web pages at www with thousands of words on each page.

The **goal of association rule mining** is to discover items that are found together in sufficient number of baskets and to find dependencies among these items. This simply implies finding frequent itemsets.

7.4.2 Handling Large Datasets for Finding Frequent Itemsets

Actual number of datasets may be very large. Finding number of disk I/Os gives the actual cost of mining the large datasets stored on the disk. The association rule algorithms read the data in iterations where all baskets read in turn. The mining cost measures using number of iterations in the algorithm over the data.

Thus, the main memory is a critical resource for several frequent itemset algorithms. The computation involves counting of occurrences of pairs when the algorithm processes the baskets. The counting of various parameters requires the usage of the main memory. Swapping the count values in/out between the main memory and the disk is not advisable.

The counting process does not eliminate useless items in later iterations, and hence wastes time without producing any useful result.

Consider the simple approach to finding frequent pairs (Similar approach can be extended for larger sets as well). It requires generating all the itemsets. Though the probability of being frequent decreases with size. It is important to count/keep track of itemsets that turn out to be frequent till the last.

The following example illustrates the method of estimating memory requirements.

EXAMPLE 7.8

Compute the memory required for finding frequent pairs in case of 1 million items.

SOLUTION

Assume an approach in which the algorithm reads the file once and counts the occurrences of each pair in the main memory. Considering n items in the basket, count process generates almost $n \cdot (n-1)/2$ pairs using two nested loops. The number of calculations involve $\sim (\text{number of items})^2$. This may result into failure if square of number of items exceed the main memory.

Memory needs for number of pairs for 1 million items compute as:

$$\Rightarrow \text{Number of pairs of items} = 10^6 \times (10^6 - 1)/2 = 5 \times 10^{11}$$

Each count value is an integer that requires 4-byte memory.

Therefore, memory needed = $4 \times 5 \times 10^{11}$ bytes = 2×10^{12} = 2 TB.

Two popular approaches for counting pairs in memory: First is to count all pairs, using a triangular matrix. This approach requires only four bytes per pair (assume integers to use 4 bytes). The second approach maintains a table of triples $[i, j, c]$ where c is the count of the pair of items $\{i, j\}$. This approach requires

12 bytes, but only for those pairs with count > 0. The second approach performs better than triangular matrix if less than one-third of possible pairs occur. This approach may also require extra space for retrieval of structure, such as a hash table.

Apriori Algorithm Section 6.5.3 discussed the algorithm. *Apriori* uses iterations (successive passes). Algorithm *Apriori* limits the need for the main memory. The first pass needs memory proportional to the number of items. The second pass needs memory proportional to the square of *frequent* items only (for counts).

Several proposed algorithms cut down on the size of candidate pairs. The following subsection describes Park, Chen and Yu (PCY), multistage and multihash algorithms [Anand Rajaraman et al.²]:

7.4.2.1 Algorithm of Park, Chen and Yu

Apriori Algorithm works efficiently when the counting of the candidate process is executing. During the first iteration, most of the memory is unused. Memory is

required to store individual item counts only. Can one use the unused memory to reduce the memory required in the second iteration?

The PCY algorithm takes benefit of the fact that the first iteration of Apriori does not use lots of main memory for counting of single items. Iteration 1 of PCY algorithm saves item counts as well as maintains a hash table with sufficient buckets that fits in memory. It also maintains the counts for each bucket into which pairs of items are hashed.

An improved version of PCY exists. Between the iterations the buckets are replaced by a bit vector. Instead of four-byte integers, one bit is used to represent the presence and absence of a frequent bucket. Bit 1 means the bucket is frequent and bit 0 means it is not a frequent bucket. Thus, the memory requirement is reduced 32 times. Also, frequent items need to be selected and listed for the second iteration.

7.4.2.2 Multistage Algorithm

A refinement of the PCY algorithm is the *multistage algorithm*. This algorithm uses several successive hash tables to reduce the number of candidate pairs subsequently. The algorithm applies more than two iterations to find the frequent pairs. The idea is to rehash only those pairs that qualify for iteration 2 of PCY after iteration 1 of PCY. Since only a few pairs contribute to buckets in the middle iteration, fewer false positives may occur. Thus, it requires 3 iterations over the data. The two hash functions have to be independent.

7.4.2.3 Multihash Algorithm

A possibility exists for getting much of the benefit of the extra iterations of the multistage algorithm in a single iteration. Multihash algorithm is an improvement of PCY. The main idea is to use several independent hash tables during the first iteration. This can lead to benefits like multistage in only 2 iterations.

7.4.3 Limited Passes Algorithms

Multistage and multihash algorithms use more than two hash functions. There is a point of reducing returns in multistage algorithm since the bitmaps mostly consume all of the main memory. The bitmaps occupy exactly what one PCY bitmap does in multihash algorithm. But too many hash functions make all

counts $\geq S_{th}$.

The algorithms for finding frequent itemsets discussed in the previous section process one iteration or pass for each size of itemset [One iteration means one pass through the sequence of instructions].

Therefore, finding itemsets of size k needs k passes. Many applications do not require finding every frequent itemset. For example, the online bookstore does not want to offer discount on all the books purchased together. Thus, they need to run an algorithm for limited number of iterations in order to find a good number of the frequent itemsets instead of all the frequent itemsets.

Simple random sampling algorithm There are certain algorithms that use two or fewer passes for all sizes. One of them is *simple random sampling algorithm*. The algorithm suggests selecting a random sample of the market baskets. Run Apriori algorithm or its improvements for sets of all sizes, not just pairs in the main memory. This does not put burden for disk I/O increase in the size of the itemsets. There should be enough space for counts while executing a simple algorithm. The algorithm reduces the support threshold proportionally to match the sample size. Thus, if the sample is $1/100$ of the total number of baskets, $s/100$ will be the support threshold instead of s . The smaller threshold facilitates more truly frequent itemsets but requires more space.

An option to implement second pass as well can validate that the sample contains truly the frequent itemsets. This avoids false positives.

7.4.3.1 SON Algorithm

An algorithm called SON (Savasere, Omiecinski and Navathe) algorithm keeps away from both false negatives and false positives using two passes.

SON algorithm repetitively read small subsets of the baskets into the main memory, and run an in-memory algorithm to find all the frequent itemsets. Subsets are not samples. It is the processing of the entire file in memory-sized chunks. An itemset becomes a candidate if it is found to be frequent in any one or more subsets of the baskets.

Second pass counts all the candidate itemsets and determine those which are frequent in the entire set. The idea of *monotonicity* used here is that an itemset cannot be frequent in the entire set of baskets unless it is frequent in at least one subset.

Distributed Version of SON also implements in a parallel computing environment. The implementation distributes the baskets among many nodes. Frequent itemsets compute at multiple nodes. The candidates then distribute to all the nodes and finally, accumulate the counts of all the candidates.

7.4.3.2 Toivonen's Algorithm

Toivonen Algorithm is similar to the simple random sample algorithm but lowers the threshold slightly for sampling. For example, if the sample s is 1% of the baskets, use $0.008 s$ as the support threshold rather than $0.01s$. The basic aim is not to miss any itemset that is frequent in the full set of baskets. As already stated, the smaller threshold results into more deserving frequent itemsets but requires more space.

After preparing the frequent itemsets for the sample, a negative border is prepared. An itemset is in the negative border when it is not considering that as frequent in the sample, but all its immediate subsets are frequent. For example, ABCD, which is not a frequent itemset, is in the negative border, if all of ABC, BCD, ACD and ABD are frequent itemsets.

Then count all the candidate frequent itemsets from the first pass and count their negative border in the second pass. If no itemset from the negative border turns out to be frequent in the second pass, then finally the candidates found to be frequent in the whole data, considered them as the frequent itemsets. The algorithm requires a restart if an itemset in the negative border is found to be actually frequent.

It suggests to choose the support threshold, which results into less probability of failure. Also, consider the number of itemsets computed on the second pass that fit in the main memory.

7.4.4 Counting Frequent Items in a Stream

The algorithms discussed in the previous subsections find frequent itemsets from a file of baskets. Now let us explore finding of frequent itemsets from a stream of baskets instead of a file of baskets.

Several large sources of data are modeled as data streams. For example, stream of network packets, sensor data, etc. It is impractical and undesirable to save and process all data exactly in such scenario. Instead, look for algorithms to find approximate answers with say one pass over data. When processing a stream,

remember that only a small part of it can be kept in the memory.

Apriori algorithm cannot be used for mining frequent patterns over a data stream. Mining using Apriori is fundamentally a set of join operations. This cannot be performed over a data stream since at any instant, a program can only examine a very limited size window of a data stream. Computation for any itemset cannot complete without considering the past and future datasets. Here, consideration is only a limited size window. That is due to the massive amount of streaming data. It is difficult to mine and update the frequent patterns in the presence of a dynamic streaming data environment.

Finding Frequent Items in Place of Itemsets

A simple frequent item finding algorithm finds all items in a stream whose frequency exceeds a $1/k$ fraction of the total count. Given a stream $S = (A, B, C, A, C, B, D, A)$, the frequency of an item a_i is f_i and total count $n = 8$. Thus $f_A = 3$, $f_B = 2$, $f_C = 2$, $f_D = 1$. For k -frequent items (if $k = 0.2$), the frequent items are the set $\{a_i \mid f_i > kn\}$.

Here, $k \times n = 0.2 \times 8 = 1.6$, therefore, frequent items are A, B, C. Similarly, for $k = 0.25$, ($k \times n = 2.0$), frequent item(s) is only A.

The frequent algorithm stores a designated number of pairs of items (say 20%) and counter for every pair of item (say a_i, c). The algorithm compares each new item against the stored items. A grouping argument is used to support the item, which occurs more than n/k times. The details are given in a research paper by Karp et al⁵.

Literature suggests that the frequent item mining algorithm sometimes does not solve the frequency estimation problem accurately. Although the algorithm preserves the bound on the true frequency of the items, it may result in some errors. Observation suggests that executing the algorithm with $k = 1/\epsilon$ implies that the count associated with each item on termination is at most $\epsilon \cdot n$ below the true value.

The other simplest approach is **randomized sampling based algorithm**. It suggests collecting some number of baskets and store them as a file. Run any one of the frequent itemset algorithms discussed in this chapter. The approach can have any two possibilities for future. Either the approach ignores the stream elements that arrive or stores them as another file, which it analyzes later. An

estimate of the frequent item sets in the stream is obtained on the process completion of frequent itemsets algorithm.

Manku and Motwani⁶⁵ proposed **Lossy Counting algorithm** in 2002. The algorithm saves the tuples consisting of an item, a lower bound on its count and a “delta” (Δ) value, which records the difference between the upper bound and the lower bound. When i^{th} item in the stream processes, if information about the item is found then its lower bound is increased by one; else, create a new tuple for the item with the lower bound set to one, and Δ set to $= \lfloor i/k \rfloor$. Also, delete at times all tuples whose upper bound is less than $\lfloor i/k \rfloor$.

Accurate values of upper and lower bounds save on the count of each item. Thus, all items whose count exceeds n/k must save at the end of the stream. As like frequent items algorithm, setting $k = 1/\epsilon$ ensures that the error in any approximate count is at most ϵn . A careful argument demonstrates that the worst-case memory space use by the algorithm is $O\left(\frac{1}{\epsilon} \log \epsilon n\right)$, and for certain time-invariant input distributions, it is $O\left(\frac{1}{\epsilon}\right)$. The algorithm details are in research paper of Manku and Motwani⁶.

7.4.4.1 Sampling Methods for Stream

Sampling is a statistical technique used for processing using a probabilistic choice of data item. The technique considers sampling in data stream and process selects a few data items from the incoming stream of data items for analysis. Section 7.3.2 explained these details.

7.4.4.2 Frequent Itemsets in Decaying Windows

Section 7.3.7 described the decaying window method for identifying the most common elements in a stream. The weight of i^{th} previous item assigns as $(1 - C)^i \approx e^{-ci}$ where $0 < (1 - C)^i \approx 1$ where $i \geq 1$. Counting frequent items in a stream requires two modifications to the algorithm for decaying windows:

1. Stream elements are baskets, and not the individual items. Maintain a weighted count for itemsets. When a new itemset arrives,
 - (i) Multiply all previous counts by $1 - C$.
 - (ii) Add a new itemset with an initial count of 1.

- (iii) Add 1 to an existing itemsets count.
2. Start counting an itemset only if all of its proper subsets are already being counted (Remember from the Apriori algorithm that if an itemset is frequent, then all of its subsets must also be frequent).

Self-Assessment Exercise linked to LO 7.3

1. How is frequent itemset analytics performed in market basket model?
2. How does the Apriori algorithm for frequent itemsets analyze?
3. How does the Park, Chen and Yu (PCY) algorithm for frequent itemsets analyze? How does PCY improve memory usages compared to the Apriori algorithm?
4. Make a table comparing the PCY, multisate and multihash algorithms.
5. List the methods for finding frequent items in data stream and compare them.

Real-time analytics platform, SparkStreaming, and real-time analytics applications to real-time sentiments analytics and stock prices analytics

7.5 | REAL-TIME ANALYTICS PLATFORM (RTAP) —SPARKSTREAMING

Real-time application relates to responsiveness. Data, when generated fast needs fast processing as well. An application sometimes requires updating the information at the same rate at which it receives data. Late decisions sometime cause loss of great opportunities. The term ‘analytics’ implies the identification of meaningful patterns from data. Thus, real-time analytics signifies finding meaningful patterns in data at the actual time of receiving it.

LO 7.4

Real-time analytics platform, SparkStreaming, and real-time analytics applications to real-time sentiments analytics and stock prices analytics

Real-Time Analytics Platform (RTAP) analyses the data, correlates, and predicts the outcomes in the real time. The platform manages and processes data and

helps timely decision-making. The platform helps to develop dynamic analysis applications. The platform leads to evolution of business intelligence.

Following are the widely used RTAPs:

1. Apache SparkStreaming—a Big Data platform for data stream analytics in real time.
2. Cisco Connected Streaming Analytics (CSA)—a platform that delivers insights from high-velocity streams of live data from multiple sources and enables immediate action.
3. Oracle Stream Analytics (OSA)—a platform that provides a graphical interface to “Fast Data”. Users can analyze streaming data as it arrives based on conditions and rules.
4. SAP HANA— a streaming analytics tool which also does real-time analytics. The SAP platform makes it easy for developers to incorporate smart stream capture and active event monitoring, alerting and event-driven response to applications.
5. SQL streamBlaze—an analytics platform, offering a real-time, easy-to-use and powerful visual development environment for developers and analysts.
6. TIBCO StreamBase—streaming analytics, which accelerates action in order to quickly build applications that analyze and act on real-time streaming data.
7. Informatica — a real-time data streaming tool which transforms a torrent of small messages and events into unprecedented business agility.
8. IBM Stream Computing—a data streaming tool that analyzes a broad range of streaming data—unstructured text, video, audio, geospatial, sensor— helping organizations spot the opportunities and risks and make decisions in real time.

7.5.1 Apache® Spark™ Streaming

Continuous arrival in multiple, rapid, time-varying, possibly unpredictable and unbounded streams have difficulties in processing, especially in Big Data with 3Vs characteristics. Streaming data processing needs computing in real time as

the data arrives.

SparkStreaming is an extension of core Apache Spark API. Spark Streaming applications are in a variety of use cases and business applications. Some of the most interesting use cases of SparkStreaming include Uber (the ride sharing service), Pinterest, (the content sharing service), Netflix (a subscription service that provides access to movies and TV shows).

SparkStreaming brings Apache Spark's language integrated API to stream processing. It facilitates building of fault-tolerant processing of streaming data in real time. SparkStreaming is one of the most popular platforms to implement data processing and analytics software for real-time data received from IoT and sensors.

Figure 5.3 showed Spark stack main components, namely *Core*, *SQL*, *Streaming*, *R*, *GraphX*, *MLib* and *Arrow* in a five-layered architecture. The architecture presented the overall Apache Spark ecosystem. All software components are also available when using SparkStreaming.

Following are the features of data processing using SparkStreaming:

1. Combines batch processing and streaming processing in the same system
2. Applies Spark's machine learning and graph processing algorithms on data stream
3. Divides the stream of data into micro-batches of a pre-defined interval (N seconds) [The N is as per need of data stream processing]. A micro-batch is used in operations similar to Resilient Distributed Datasets (RDDs). A very low value of N means that the micro-batches may not have sufficient data for useful analysis.
4. Support Scala, Java, R and Python language in the form of suitable APIs
5. Results save at a data store for performing analytics later
6. Results also generate reports, display visuals on live dashboard and generate alerts on the events.

SparkStreaming library is used for processing a real-time data stream:

1. **DStream** (Discretized Stream)—an abstraction of a continuous data stream in SparkStreaming DStream creates either from basic sources or input data

stream from sources, such as Kafka, Flume and Kinesis. Stream operators apply functions on DStreams. DStream represents streaming data from a TCP source.

2. **Type of Input Sources**—(i) Basic sources: Sources which are directly available in the *StreamingContext* API, such as a file system or a socket connection, and (ii) Advanced sources which are available from sources such as Kafka, Flume, Kinesis.
3. **Apache Kafka**— is a real-time, fault tolerant, scalable messaging system for moving data in real time. Kafka captures user activity on websites, logs, stock ticker data and instrumentation data. Kafka works like a distributed database and is based on a partitioned and replicated low latency commit log. Apache Kafka includes client API as well as a data transfer framework called Kafka Connect.
4. **ZooKeeper**—a centralized service providing reliable distributed coordination for distributed applications. Kafka, the messaging system for configuration details across the cluster.

Figure 7.7 shows various architecture components of SparkStreaming for Big Data applications, analytics, live data visualization, real-time online recommendations and instant fraud detection.

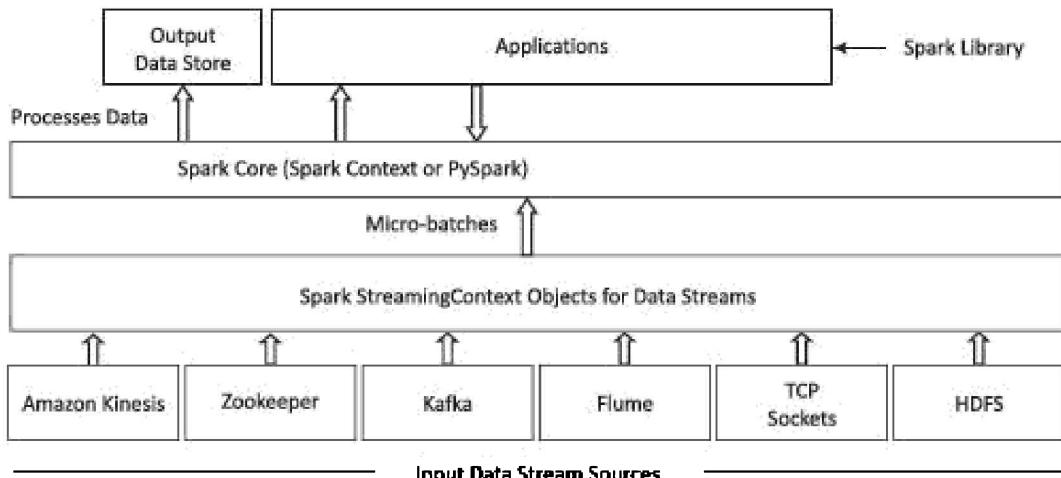


Figure 7.7 Various architecture components of SparkStreaming application

Table 7.2 gives the functionalities and operators of SparkStreaming

Table 7.2 Brief description of SparkStreaming functionalities and operators

Operator	Brief Description
Creating StreamingContext Object using SparkConf	
import org.apache.spark._ import org.apache.spark.streaming._ val conf = new SparkConf(). setAppName (appName). setMaster(master) val ssc = new StreamingContext(conf, Seconds(1))	Creates StreamingContext object named ssc configured with AppName; after the creation of ssc, subsequent steps define the (i) input sources by creating input DStreams, and (ii) the streaming computations by applying transformation and output operations to DStreams.
Operations on DStreams from input sources	
union (otherStream)	Return a new DStream that contains the union of the elements in the source DStream and otherDStream.
join(otherStream, [numTasks])	When called on two DStreams of (K, V) and (K, W) pairs, return a new DStream of (K, (V, W)) pairs with all pairs of elements for each key.
Cogroup (otherStream, [numTasks])	When called on a DStream of (K, V) and (K, W) pairs, return a new DStream of (K, Seq [V], Seq [W]) tuples.
Streaming transformation operations	
transform (func)	Return a new DStream by applying a RDD-to-RDD function to every RDD of the source DStream. This can be used to do arbitrary RDD operations on the DStream

Stream APIs Processing RDD like Operations

<code>map(func):</code>	Return a new DStream by passing each element of the source DStream through a function func.
<code>flatMap(func):</code>	Similar to map, but each input item can be mapped to 0 or more output items.
<code>filter(func)</code>	Return a new DStream after selecting the records of the source DStream on which func returns only true.
<code>reduce(func)</code>	Return a new DStream of single-element RDDs by aggregating the elements in each RDD of the source. DStream uses the function func (which takes two arguments and returns one). The function should be associative and commutative so that it can be computed in parallel.
<code>repartition(numPartitions)</code>	Changes the level of parallelism in this DStream by creating more or fewer partitions
<code>count()</code>	Count the number of elements in each RDD of the source DStream and return a new DStream of single-element RDDs.
<code>countByValue()</code>	When called on a DStream of elements of type K, return a new DStream of (K, Long) pairs where the value of each key is its frequency in each RDD of the source DStream.
<code>reduceByKey(func, [numTasks])</code>	Return a new DStream of (K, V) pairs when called on a DStream of (K, V) pairs. The values for each key are aggregated using the given reduce function.
<code>updateStateByKey(func)</code>	Return a new “state” DStream where the state for each key is updated by applying the given function on the previous state of the key and the new values for the key. This can be used to maintain arbitrary state of data for each key.

Streaming output operations

<code>foreach RDD(func)</code>	‘foreach’ is the most generic output operator that applies a function, func, to each RDD generated from the stream. This function should push the data in each RDD to an external system, such as saving the RDD to files, or writing it over the network to a database.
<code>saveAsTextFiles(prefix, [suffix]):</code> <code>saveAsObjectFiles(prefix, [suffix])</code> <code>saveAsHadoopFiles(prefix, [suffix])</code>	<ul style="list-style-type: none"> (i) Save the DStream contents as text files. The file name at each batch interval is generated based on prefix and suffix: “prefix-TIME_IN_MS [suffix]”. (ii) Save the DStream contents as SequenceFiles of serialized Java objects. The file name at each batch interval is generated based on prefix and suffix: “prefix-TIME_IN_MS [suffix]”. (iii) Save the DStream contents as Hadoop files. The file name at each batch interval is generated based on prefix and suffix: “prefix-TIME_IN_MS [suffix]”
<code>print()</code> or <code>pprint()</code> in Python	Prints the first ten elements of every batch of data in a DStream on the driver node running the streaming application; This is useful for development and debugging.

Stream APIs (time) Window Related Transformation Processing Operations

Window(windowLength, slideInterval) countByWindow(windowLength, slideInterval) reduceByWindow(func, windowLength, slideInterval) reduceByKeyAndWindow(func, windowLength, slideInterval, [numTasks]) reduceByKeyAndWindow(func, invFunc, windowLength, slideInterval, [numTasks])	These transformation operations are performed for the stream of data during SlidingInterval for the time equal to windowLength. <ul style="list-style-type: none">• WindowLength specifies the duration of the window.• Sliding interval specifies the interval at which the window operation is performed.
Starting data receiving and processing	
streamingContext.start()	Processing starts
Termination and stop operations	
streamingContext.awaitTermination()	Wait for the termination signal (CTRL+C or SIGTERM) from user to stop streaming process.
streamingContext.stop()	Stop streaming process immediately.

The following example shows use of SparkStreaming for count functions.

EXAMPLE 7.9

Create a program in Scala for counting the number of words from a socket in every 1 second, using window of length 60 seconds and sliding interval of 10 seconds, and save the result in windowed WordCounts. Use SparkStreaming context and API.

SOLUTION

- (i) Refer Table 7.2. Import *SparkStreaming* using program statements as follows:

```
import org.apache.spark._  
import org.apache.spark.streaming._
```

- (ii) Configure *appName* and *StreamingContext* using program statements as follows:

```
val conf = new  
SparkConf().setAppName(StreamWordCount).setMaster(master)  
val ssc = new StreamingContext(conf, seconds(1))
```

Here, ssc stands for SparkStreamingContext.

- (iii) Create a *socketTextStream* at ip:port for count the words as follows:

```
// Replication necessary in distributed scenario  
for fault tolerance.
```

```
val lines = ssc.socketTextStream(args(0),  
args(1).toInt, StorageLevel.MEMORY_AND_DISK_SER)
```

- (iv) Compute streamWordCount using *map()* and *reduce()* as follows:

```
val streamWords = lines.flatMap(_.split(" "))  
val windowedWordCounts = streamWords.map(x => (x,  
1)).reduceByKeyAndWindow(_ + _, Seconds(60),  
Seconds(10))
```

- (v) Start the processing, and await termination using the program statement as follows:

```
ssc.start()  
ssc.awaitTermination()
```

- (vi) Print the results using the program statement as follows:

```
windowedWordCounts.print()
```

7.5.2 Real-Time Analytics Platform Applications

Some such applications are:

1. Fraud detection systems for online transactions
2. Log analysis for understanding usage pattern
3. Click analysis for online recommendations
4. Push notifications to the customers for location-based advertisements for retail
5. Action for emergency services such as fires and accidents in an industry
6. Any abnormal measurements require immediate reaction in healthcare

monitoring

7. Social media.

7.5.3 Case Studies—Real-Time Sentiment Analysis, Positive Negative Sentiments Prediction and Stock Market Predictions

Real-time data feeds from social media (such as twitter) are easy to get. A use of this data is for sentiment analysis. The real-time data feeds of stock prices at stock trading exchange are available. A use of these feeds is in sentiment analysis and future price predictions.

The following subsections considers these examples for real-time analysis and predictions.

7.5.3.1 Real-Time Sentiment Analysis using Tweets

The case study provides the method of access of real-time social media information using Twitter. Tweets are received from the Twitter stream, pre-processed and then analyzed to extract the features. The method follows the steps as:

1. Collect a comprehensive training dataset that consists of data about potential users of a system.
2. Pull the specific tweets in real-time using Twitter API, then process and load this data into a persistent storage.
3. The cleaning of the data proceeds with punctuations, stop words, URLs, common emoticons and hash tags, references deletion. Multiple consecutive letters in a word are reduced to two (words like tooooooooooo much... is replaced with too much). Spell checking is also performed to words that have been identified as misspelled in order to infer the correct word.
4. Classify various types of tweets and segment them after estimating the influence of each tweet using the predictive analysis library. Perform sentiment analysis by identifying whether people are tweeting positive or negative statements about some actions.

5. Use linguistic concepts, perform opinion mining, analyze data and bring out powerful insights. Use important feature of the analysis based on machine learning. Thus, applications learn by analyzing ever-increasing amounts of data.
6. The goal is to build the model for predicting the sentiments from tweets. Table 7.3 presents sentimental analysis features.

Table 7.3 Sentiment analysis features

Feature	Meaning
NEGATION	Presence of negating words
POSITIVE SMILEY	Presence of common positive emoticons
NEGATIVE SMILEY	Presence of common negative emoticons
DONT— YOU, OH, SO, AS FAR AS,	May indicate ironic or sarcastic text
LAUGH	Presence of popular laughter indications, such as haha, lol

Tweets sentiments prediction can face the following problem: The use of negatives and positives in the same sentence. For example, “I like red color but I hate blue color”. A classification of such sentence is that it is a neutral sentence. The application handles that sentence by breaking the sentence into two subparts such that one is positive and the other is negative. The count value thus does not alter.

This classification model adapts to the evolution of tweets and employs the user (or domain expert) feedbacks.

The outcomes affect by:

1. Varying size of dataset
2. Different features
3. Rate with which tweets arrive
4. Changes in the textual content (for example, the changes in vocabulary, meaning of words, etc.)

Social media sentiment analysis also needs to identify various human emotions like sadness, anxiety, fear, confusion, depression and anger.

7.5.3.2 Stock Market Predictions

Stock market data is an example of a real-time data stream. Data stream algorithms compute the values over a time-window of stock trades. This window has fixed size and contains n stock trades. A sale-purchase is counted as one trade. The parameter studied is Volume-Weighted Average Price (VWAP).

The data stream model can be relational tuples-based model with tuples (sale_time, ticker_symbol, num_shares, price_per_share) for each stock sales in real time. Assume stock trade i has price P_i , with S_i shares changing hands, then compute the volume-weighted average (VWAP) stock prices from stream of stock sales. P_{VWAP} (Price for VWAP) = $\sum P_i S_i \div \sum S_i$.

The algorithm first transforms stock-sale data stream into a relation using a time-based sliding-window operator with sliding interval $\Delta T = 5$ m (over last 5 minutes) of data. The window length is time from start of the day and thus increases as the day progresses.

The algorithm estimates value of P_{VWAP} continuously, and evaluates the relation factor r periodically during trading day.

Recapitulate equation (6.8a). The correlation coefficient r between two variables x and y is:

$$r = [1 / (n - 1)] \times \sum \{ [(x_i - \bar{x}) / s_x] \times [(y_i - \bar{y}) / s_y] \}, \quad \dots (7.3)$$

where n is the number of observations in the sample. x_i is the x value (= time since start of trade in a day) for i^{th} observation. \bar{x} is the sample mean of x values. y_i is the y value ($= P_{VWAP}$) for i^{th} observation. \bar{y} is the sample mean of y values. s_x is the sample standard deviation of x. s_y is the sample standard deviation of y.

Compute a correlation factor (relation) containing aggregates using standard grouping/aggregation operations. The r is also sent as another stream (results stream).

Refer Sections 6.2.2 to 6.2.4. A prediction model can be building a relationship r with respect to time and then predict using machine learning tools. Value of $r > 0$ continuously over a window of time length, (for example, 4 hours) indicates a continued positive relationship (sentiment towards the stock price). Conversely, $r < 0$ indicates a negative relationship (sentiment) and $r = 0$ indicates no relationship (or that the variables are independent of each other and not

related). A prediction for price increase or decrease can be based on positive or negative sentiments during the period of study from the start of the trade.

Self-Assessment Exercise linked to LO 7.4

1. List the applications of real-time analytics.
2. List the platforms available for real-time analytics.
3. How are the features in SparkStreaming used for data stream analysis?
4. Explain transformation functions in SparkStreaming.
5. How is real-time data stream used in making predictions?



KEY CONCEPTS

adapter

Apache Spark

Apriori algorithm

association rule

BDAS

Big Data

Bloom filter

correlation coefficient

COUGAR

counting distinct

counting ones

CQL

data stream

DBMS

decaying window

DSMS

DStream

filtering
frequent itemset
hash function
HDFS
Kafka
lambda architecture
moments
multihash algorithm
multistage algorithm
operator
predictive analytics
queries on data stream
real-time analytics
real-time data
relational tuple
representative sample
sample size
sampling
sentiments analysis
sliding window
StreamingContext
TelegraphCQ
tuple
window



LO 7.1

1. A stream is a sequence of data elements or symbols made available over time. Transmitting or receiving (data) between computer systems or networks is streaming of data. Mostly, data stores process data using batched processing. Processing streaming data is different from processing the data saved at a store.
2. Stream processing uses graph-based data stream model, relation-oriented stream tuples model, object-based or windows-based model.
3. Data stream processing architecture is as follows: Queries are required to be processed on streaming data. Applications continuously handle queries from a query repository. Streaming data is processed after load sharding at the memory. The queries response saves at the output buffer before finally being retrieve in the application.
4. Data Stream Management System (DSMS) manages streaming data.
5. Issues in stream processing are large data stream from different domains, variation in frequency of data stream, zeros, unbounded sizes, need of scalable, near-real time or event-based processing and need of filtering undesirable data.

LO 7.2

1. Stream computing uses algorithms which analyze the data in real time at high speed and accuracy. Stream computing is the fastest and most efficient way to obtain useful knowledge from Big Data analytics. Organizations react quickly on appearance of a problem and can predict new trends for the future.
2. Sampling in data stream means the selection of a few data items from incoming stream of data items for analysis. Choice of data items for processing is as per probabilistic sampling.
3. Several filtering techniques exist: Bloom Filter and its variant, Streaming Quotient Filter (SQF), Particle filter, Kalman filter, XML filters (XFilter,

YFilter) are some of the popularly known stream filters. Bloom filter does conditional matching where a tuple matches with some desired value. The Bloom filter is simple to implement and space efficient.

4. The stream of data contains repeated elements. Counting distinct elements find the number of dissimilar elements in a data stream. This has applications in the area of networking and databases.
5. The sliding window model for data stream algorithms is the one in which data elements receive one by one and statistically compute over a sliding window of size N (not over the whole stream). The window covers the most recent data items arrived.
6. Decaying windows technique computes a smooth aggregation of all the 1s ever seen in the stream, with decaying weights. When it further appears in the stream, less weight is assigned.

LO 7.3

1. Finding frequent itemsets means finding associated items in sets which are found together sufficiently in number of baskets and to find dependencies among the items.
2. Apriori algorithm for frequent itemsets works efficiently when the counting of the candidate process is executing.
3. PCY algorithm takes benefit of the fact that in the first iteration of Apriori, the counting of single items does not require lots of main memory. Multistage algorithm uses several successive hash tables to reduce the number of candidate pairs subsequently. Multihash algorithm uses several independent hash tables on the first iteration. This can lead to benefit like multistage in only 2 iterations.
4. SON algorithm repetitively reads small subsets of the baskets into the main memory and runs an in-memory algorithm to find all the frequent itemsets.
5. Frequent algorithm helps in finding all the items in a stream. It finds the associated sets whose frequency exceeds a $1/k$ fraction of the total count.

Lossy counting algorithm stores tuples consisting of an item, a lower bound on its count and Δ value which records the difference between the upper and lower bounds

6. Counting frequent items in a stream requires two modifications in the decaying window algorithm.

LO 7.4

1. Real-time application relates to responsiveness. When data is generated fast, it needs fast processing as well.
2. Apache SparkStreaming and several tools enable real-time analytics. SparkStreaming is an extension of core Apache Spark API. SparkStreaming brings Apache Spark's language-integrated API to stream processing.
3. SparkStreaming provides a number of transformation functions.
4. Real-time analytics with regression analysis, statistical functions and machine-learning tools can predict positive, negative or no correlations in real time from the stream of data for applications such as stock prices.

Objective Type Questions

Select one correct answer option for each of the following questions:

- 7.1 A stream processing engine may include (i) the core low-latency stream processing functionality with (ii) a rich set of stream-oriented operators, (iii) coordinator, (iv) loader, (v) manager, (vi) load shedder, (vii) fault tolerance module, (viii) graphical query editor, (ix) system visualizer, and (x) stream connection generator.
- (a) all except viii to x
 - (b) all
 - (c) all except iii, vi and viii
 - (d) i, ii, ix and x

7.2 DBMS (i) stored sets of records with no pre-defined time concept, (ii) is suitable for applications that require persistent data storage and complex querying, (iii) sequence of data elements, (iv) one-time queries, and (v) bounded main memory.

Data Stream Management System (vi) provides online analysis of rapidly changing stream of data, (vii) is suitable for real-time, continuous, ordered (arrival time or timestamp) (viii) persistent relations (relatively static, stored), (ix) transient stream (on-line analysis), (x) implements sequential access, and (xi) unbounded disk storage.

- (a) none
- (b) all except iii, v, viii and xi
- (c) only ii
- (d) all

7.3 Data stream model for processing can be based on (i) windows, (ii) relation-oriented tuples, (iii) correlation, (iv) graph, and (v) queries.

- (a) i, ii and iv
- (b) all
- (c) i to iii
- (d) all except iv

7.4 Stream processing issues are (i) unfixed size stream, (ii) unbounded data, (iii) need of scalable processing, (iv) variation in frequency of data stream, (v) may need real-time processing, and (vi) large data streams from different domains.

- (a) all except ii, iii, iv, xii and xiii
- (b) all
- (c) all except ii to vi
- (d) all

7.5 Minimum sample size required for accuracy in estimating proportions, the following are taken into the consideration: (i) A precise estimate of key

proportions P to be measured in the study.

(ii) The degree of error D that is desired in the study, ~1%-5% or 0.01 and 0.05, (iii) the confidence level $Z = 95\%$ needed from the inference, and (iv) null hypothesis true.

(a) i, iii, iv

(b) i to iv

(c) iii

(d) All except iv

7.6 Process of deleting a particular element in the Bloom filter requires that (i) the corresponding positions computed by k hash functions in the bit vector be set to 1. In order to perform deletion of the element, the concept of the counting Bloom filters as a variant of Bloom filter functions as follows: (ii) The counting filter maintains a counter for each bit in the Bloom filter. (iii) The counters corresponding to the k hash values are incremented or (iv) decremented, whenever an element in the filter is deleted. (v) As soon as a counter changes from 0 to 1, the corresponding bit in the bit vector reset to 0.

(a) i, ii, iv

(b) ii to v

(c) ii and iv

(d) All except iv

7.7 The sliding window model for data stream algorithms is for (i) infinite, (ii) bounded data stream processing. The window refers to (iii) time interval (iv) number of data items during which stream raised the queries and processed. (v) The data elements are received one by one and the statistics are computed over a sliding window of size N (not over the whole stream). (vi) The window covers the last arrived data items.

(a) i, iii and v

(b) all

- (c) ii to iv
- (d) i, ii, iv, v

7.8 The goal of association rule mining is (i) to discover items that are found together in (ii) sufficient number of baskets and (iii) to find dependencies among these items. (iv) This simply implies finding the frequent itemsets. (v) Rule should define an itemset present in a certain percentage of baskets, then that set is a frequent itemset. (vi) It is not necessary that an itemset is present in a basket only once. (vii) Itemset presence is counted once though the itemset but it may be present 2, 3 or more times in a basket.

- (a) all except ii, iii and v
- (b) all
- (c) all except vi
- (d) i to iv

7.9 Multistage algorithm uses (i) several successive hash tables to reduce the number of candidate pairs subsequently. (ii) The algorithm applies more than two iterations to find the frequent pairs. (iii) The idea is to rehash only those pairs that qualify for iteration 3 of Park, Chen and Yu (PCY) algorithm after iteration 2 of PCY. (iv) Only a fewer pairs contribute to buckets in the middle iteration, (v) so fewer false positives may occur. (vi) It requires 3 iterations over the data. Iteration 3 does (vii) count only those pairs {i, j} that satisfy the certain candidate pair conditions and (viii) both i and j are frequent items. Conditions are (ix) the pair uses the first hash function to a bucket whose bit in the first bitmap is 1. (x) The pair uses the second hash function to a bucket whose bit in the second bitmap is 0.

- (a) all except v and vii
- (b) all
- (c) all except iii and x
- (d) i to vii

7.10 Consider frequent itemsets finding problem use (i) decaying window method for identifying the most common elements in a stream. (ii) The

weight of i^{th} previous item assigns as $(1 - C)^i \approx e^{-ci}$, (iii) relation $0 < (1 + C)^i \approx 1$ exists, (iv) value of $i \geq 1$, and (v) start counting an itemset only if all its proper subsets already being counted.

- (a) all
- (b) all except v
- (c) all except ii to iv
- (d) all except iii and iv

7.11 A real-time application relates to responsiveness as soon as (i) a data source sends the data,

(ii) the data stores in memory, (iii) the data generates and (iv) the data that is being generated fast must first be saved and then processed fast at any time. (v) An application sometimes requires updating information at the same rate as it receives data. (vi) Late decisions sometime lead to loss of great opportunities.

- (a) all except ii and iv
- (b) all
- (c) all except ii, iii and iv
- (d) ii to v

7.12 When making the real-time sentiments analysis for making predictions (i) a correlation coefficient (relation) r computes in real time, (ii) the coefficient r accounts for the aggregates using standard grouping/aggregation operations, (iii) The r is also sent as another stream (results stream).

(iv) A prediction model can be first built using machine learning tools after building a relationship r with respect to number of frequent sets. (v) Value of $r < 0$ continuously over a window length indicates a continued positive relationship (sentiment towards the stock price). (vi) Conversely, $r > 0$ indicates a negative relationship (sentiment) and (vii) $r = 0$ indicates no relationship (or that the variables are independent of each other and not related).

- (a) all except ii, iv and xii

- (b) all except iv, v and vi
- (c) all except vii
- (d) all except ii and iv

Review Questions

- 7.1 Describe various data stream models for extracting knowledge structures from a continuous stream. Give reasons for using each of these models. **(LO 7.1)**
- 7.2 Describe Data Stream Management System. How does it differ from a DBMS? **(LO 7.1)**
- 7.3 Describe the difficulties in real-time data stream analytics. How are they solved? **(LO 7.1)**
- 7.4 List the approaches for calculating the sample size. What are the parameters considered for calculating the minimum sample size required for accuracy in estimating proportions? **(LO 7.2)**
- 7.5 How does a stream filter function? Describe Bloom filter. **(LO 7.2)**
- 7.6 Describe an algorithm to count the distinct number of dissimilar elements from data stream. **(LO 7.2)**
- 7.7 How do different algorithms find the associated items in sets, which are together in sufficient number of baskets? How do you find dependencies among these items? **(LO 7.3)**
- 7.8 How is the frequent items counting done in a stream? Describe the different methods used.
(LO 7.3)
- 7.9 What are the types of applications in which Real-Time Analytics (RTA) enables timely decisions? What are the tools used by the RTA platform? **(LO 7.4)**
- 7.10 Make a diagram for SparkStreaming computing architecture components.
(LO 7.4)

7.11 What are the features in Spark and SparkStreaming for stream computation on Big Data? **(LO 7.4)**

7.12 What are DStreams? What are the functions used for transformation and processing of a DStream? **(LO 7.4)**

Practice Exercises

7.1 A file has data of 1000 rows similar to table given below:

Table of Product categories, ProductId, and Product name

ProductCategory	ProductId	ProductName
Toy_Airplane	10725	Lost Temple
Toy_Airplane	31047	Propeller Plane
Toy_Airplane	31049	Twin Spin Helicopter
Toy_Train	31054	Blue Express
Toy_Train	10254	Winter Holiday Toy_Train

How will you use the relation-based data stream model? Assume no time-stamping of the data. **(LO 7.1)**

7.2 Summarize the commonalities and differences in data stream query languages, (i) Relation based—CQL (STREAM), StreaQuel (TelegraphCQ), (ii) Object-based: Tribeca or ADT model-based sources COUGAR and (iii) Procedural-based Aurora in which a user specifies the data flow. **(LO 7.1)**

7.3 Describe steps for developing the algorithm for various data stream filtering algorithms.
(LO 7.2)

7.4 List the steps for developing the algorithm for various data stream counting of the distinct elements. **(LO 7.2)**

7.5 List the steps in algorithms Apriori, PCY, multistate and multihash methods of frequent itemsets analytics of data stream. **(LO 7.3)**

7.6 Write steps in computing the frequent itemsets in data stream using the decaying window method. **(LO 7.3)**

7.7 Create 80 exemplary tuples (sale_time, ticker_symbol, num_shares, price_per_share) on every five minutes from the start of stock trading at 9 am to 1 pm on the relation-oriented stream tuples model. Choose your own stock. (Can use <http://www.moneycontrol.com> charts for stock quotes during a day. Write the code in Python or Java, compute and plot correlation coefficient and stock prices as a function of time from the start of the stock trade. **(LO 7.4)**

1 B. H. Bloom, Space/time trade-offs in hash coding with permissible errors. Communications of the ACM, 13(7):422– 426, 1970.

2 Anand Rajaraman and Jeffrey David Ullman in their book “Mining of Massive Datasets”, Cambridge University Press, 2012.

3 Flajolet, P. and Martin, G. N, “Probabilistic counting algorithms for data base applications”, Journal of Computer and System Sciences, 31(2), 182–209, 1985

4 <http://algo.inria.fr/flajolet/Publications/FlMa85.pdf>

5 Karp, R., Papadimitriou, C., Shenker, S. “A simple algorithm for finding frequent elements in sets and bags”, ACM Transactions on Database Systems, Volume 28, Pp. 51–55, 2003

6 G. Manku and R. Motwani, “Approximate frequency counts over data streams”, International Conference on Very Large Data Bases, pages 346–357, 2002.

Note:

○○● Level 1 & Level 2 category

○●● Level 3 & Level 4 category

●●● Level 5 & Level 6 category

Chapter 8

Graph Analytics for Big Data and Spark GraphX Platform

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- LO 8.1 Model the database as graphs, and represent the graphs using triples
- LO 8.2 Get knowledge of graphs, graph network-organization, choose the graphs for analytics, and know graph-analytics use cases
- LO 8.3 Get conceptual understanding of graph parameters, methods, diagnostics and decisions, statistical model, StatsModel, probabilities-based graph-analytics, and understanding of technical complexities in analyzing the graphs
- LO 8.4 Use the Apache Spark GraphX, a Big Data graph-analytics platform. Know the features, architecture and components. Apply them for graph-analytics

RECALL FROM EARLIER CHAPTERS

A Big Data store system is HDFS (Section 2.3). Big Data store uses NoSQL format datasets. NoSQL data do not model like relational tables. BigData analytics algorithms process the NoSQL format datasets.

Graph databases can model the NoSQL databases also. (Section 3.3.5). A graph database consists of edges which interconnect the data nodes (vertices). The interconnections represent the relationships, associations and properties.

Apache Spark includes GraphX. Graph analytics tasks execute with ease using Spark GraphX. GraphX extends the Spark and thus has RDD (Resilient Distributed Data) property. The API consists of a collection of graph algorithms for analytics. Computations in GraphX use fundamental operators (such as subgraphs, joinVertices and aggregateMessages). (Section 5.2)

This chapter focuses on graph databases, organization of graph networks and the graph analytics platform, GraphX.

8.1 | INTRODUCTION

'Graph' is a set of vertices and edges. Graph theory is the theory of graphs and their properties. Examples of graphs are *hierarchy* graph, *monotone* graph, *connected* graph, *bipartite* graph, *planer* graph and *triangle free* graph.

Graphs have a number of characteristics. A graph:

1. *Represents a database* using graph parameters or properties assigned to each vertex, v and edge, e . An edge is a line joining two vertices. Graph nodes and edges connect each other through relations, associations and properties.
2. *Represents an abstract data type* for the relationships. A graph depicts relationships, such as, a relationship between two or more quantitative dependent variables with respect to an independent variable.
3. *Represents knowledge and reasoning* in a conceptual graph model
4. *Represents a network*, such as a social network
5. *Models the dataflows and programflows* using directed graphs: A Dataflow Graph (DFG) represents the flows of data and program. The DFG consists of sets of circles. A circle represents a node (vertex). Each node represents a set of computations or a set of operations which change the *initial state* to a *new state* (of entities or properties). State change occurs on receiving new inputs followed by computations at a node. The incoming directed edges represent the inputs received from the other nodes. The outgoing directed edges represent the output to the other nodes.
6. *Computes using path traversal*, which means going through a finite or infinite sequence of edges in a graph that connect a sequence of vertices between initial vertex v_0 and end vertex v_e . Traversing is along a path from v_0 to v_e along the connected edges.
7. *Analyzes data using graphic parameters*, relationships, associations and distribution of properties along the vertices in the path, and property variations on path traversal from a node to other nodes along the edges
8. *Analyzes data through the queries* on data using path traversal, which means from v_0 following the sequence of steps to v_e .

This chapter describes graph models, graph parameters, graph network organizations and graph analytics. It also describes statistical models for changes and distribution of properties on path traversal between the nodes. Section 8.2 describes modeling of databases as graphs and representations of graphs using triples. Section 8.3 describes graphs and graph networks. The section also describes choosing of a graph for analytics, and use cases of graph analytics. Section 8.4 describes graph parameters, methods of diagnostics and decisions, StatsModel, probabilities-based analytics, and technical complexities in analyzing the graphs. Section 8.5 describes the features of Apache Spark GraphX, its architecture, components, applications, and the considerations of using the dedicated appliances for the graphs.

8.2 | GRAPH MODEL

A graph represents *an abstract data type*. The edges of a graph represent relations, connections or associations. The vertices represent the entities. Each entity can have parameters assigned to that. Each

node can have parameters and property assigned to that.

A set of vertices (nodes) V and edges (links) E define a graph G . Relation in terms of set theory is $G = (V, E)$, which means that graph G is a set, which contains two sub-sets, vertices V and edges E .

1. Elements of V represent the entities. A node or a vertex v in set V represents an entity, such as `studentID` [Example two nodes: 'studentID' and 'studentSem1SGPA'].
2. Elements of E represent the relations or associations. An edge connects the two nodes. An edge, e represents a relation or association between the two entities. An example of association is 'studies at'. Student of an ID studies at a UG course where the `studentID` and UG course are two entities at two interconnected nodes.

Order of a graph specifies by number of vertices N_v and number of edges N_e , where $N_v = |V|$, and $N_e = |E|$.

The degree of a node (node degree) specifies the number of edges linked to a node. The degree may vary from traversing from one node to the other. For example, three linkages $v1$ to $v2$, $v1$ to $v3$, $v1$ to $v4$ mean that $v1$ (node-degree) is 3. A distribution function represents the variation in degrees of the nodes on traversing (Section 6.2.5).

A graph model has the following features:

- (i) A label near an edge can specify the context of relation or association. A label at an edge can also specify a value. For example, the grade point average in Semester 1, `studentSem1SGPA`.
- (ii) A label near a vertex can specify identification for the entity. For example, `studentID`.
- (iii) A weight near the edge can specify the weight of a relationship with respect to other edges of the same kind.
- (iv) A property can associate with the vertex or edge. For example, `adding` property.
- (v) Multiple relationships can associate a pair of vertices interconnected by multiple edges.
- (vi) A direction can associate with the direction of flow of relationship or association.

A graph, called *property-graph*, consists of each vertex and edge assigned properties (Section 8.2.5).

A graph, called *directed graph*, consists of directed edges. A directed graph shows a sequence of edges (or arcs) which connect a sequence of vertices with a condition. All directed edges are in the same direction when a graph has edges directed inwards toward a node and outwards towards another node, such that inward edge(s) represent an input for computation or state change at the node and outward edge(s) represent the output(s) which is input to the next node in the graph.

The number of inward edges in a directed graph is a node parameter called *in-degree*. Directed graph node *out-degree* means the number of outgoing edges from the node. A directed graph represents the flow of the relationship using the directed edge. For example, assume Semester 1 and SGPA are vertices in a directed graph. The edge represents a relation between them, i.e., it represents that SGPA is the result of the examination of Semester 1.

A graph defines the entities and properties to each vertex and edge. A graph called *directed multigraph*, provisions for the multiple parallel edges and that enables multiple relationships between the entities. Multiple parallel edges share the common source and destination vertices. *Directed Acyclic Graphic* is a

special kind of directed graph that contains no cycles.

Consider columnar data store in a tabular representation. Each row-group consists of a sequence of columns. Relationships between columns of the row groups are implicit (for search and queries for the values in columns) but not explicitly specified (Section 3.3.3).

Graph database explicitly stores the relationships at each edge. A hierarchy graph stores hierarchical relationships. Hierarchy relations between the tables do not store but implicit in the codes for a search or query.

The following example explains the usages of nodes, edges and properties in a graph model. The example gives a corresponding tabular Data Store.

EXAMPLE 8.1

Consider the students, that are studying Bachelor of Computer Science course and have appeared in Semester 1 examination in the department.

- (i) How does a graph model show the relationships and semester grade point averages?
- (ii) How does a table of grade point averages (GPAs) of a student in a departmental semester examination correspond to the graph model?

SOLUTION

- (i) Figure 8.1 shows a graph model for a student grade-sheet database. The graph consists of nodes and edges for a first-semester UG computer-science student of ID = UGCS4268 . The figure shows the relationships using label and property at vertices and edges for a student.

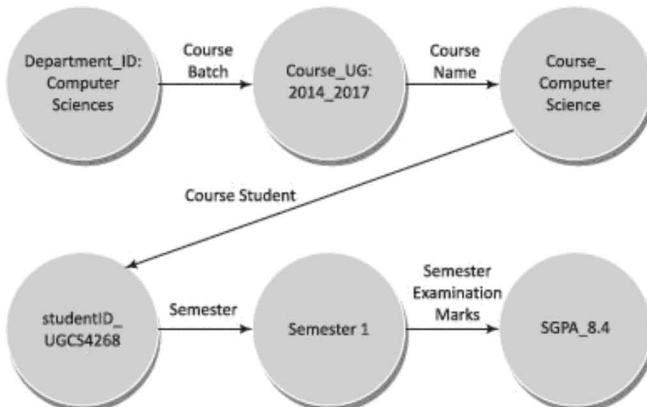


Figure 8.1 Graph model of a grade sheet

- (ii) Tabular representation in an RDBMS mapped with the above graph model for the Semester Grade Point Averages (SGPAs) is as follows:

Department_ID	Course Batch	Course Name	Student ID	Semester	SGPA
Computer Sciences	UG 2014_2017	Bachelor in Computer Science	4268	1	8.4
...
...

8.2.1 Representing a Graph as Triples

Triple means a data entity consisting of three-components: subject, predicate and object. For example, consider a sentence, ‘Spark includes GraphX’. Here, Spark is the subject, includes is the predicate and GraphX is the object.

Triples represent the Graph entities. Assume a directed graph. A triple is a sentence-like format. A sentence consists of three elements, ‘subject’ ‘predicate’ and ‘object’. Similarly, a triple consists of three elements: *source node* (subject) connects to *destination node* (object) through an *edge* (predicate).

Triple has a subject-predicate-object format for representing three elements: source node, edge and destination node. Format is *instance identifier-property name-property value*. For example, StudentID: 42629 obtained: “GradePoint_Java “8.2”. StudentID: 42629 is the instance identifier. Obtained is the property name. Value of Property GradePoint_Java is 8.2. Graph model represents instance identifier and property value at two nodes and property name as the edge connecting them.

A graph node defines by a vertex such that the two of them connect through a relationship or association. For example, consider a sentence, ‘Raj Kamal wrote textbook on Internet of Things’. ‘Raj Kamal’ is a node and ‘textbook on Internet of Things’ is next node.

A graph edge is an interconnecting line or arrow, which represents a relationship or association in a sentence. Consider another example of triple consisting of a subject, verb, and object in a sentence, ‘Raj Kamal wrote a classic book on Embedded Systems’. Verb ‘wrote’ relates the *subject* ‘Raj Kamal’ and *object* ‘classic book on embedded system’. Two vertices are subject and object in the graphical representation. The edge joining them represents the *verb*.

Following explains the representation of a graph model Data Store as triples:

EXAMPLE 8.2

Recapitulate Example 1.6(i). Consider the sales figures of Kit Kat, Milk, Fruit and Nuts, Nougat and Oreo. (i) Show a graph model for database of yearly sales and (ii) write the triples, which represent the graph.

SOLUTION

- (i) Figure 8.2 shows a graph model for nodes and edges showing the relationships of yearly sales of chocolates.

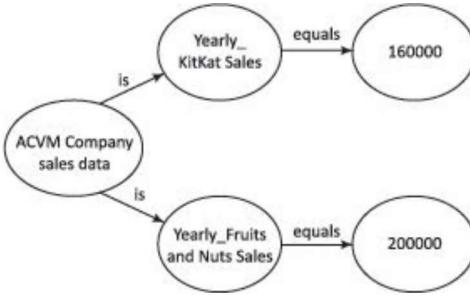


Figure 8.2 Example of a graph model of yearly total sales of an ACVM company

- (ii) Graph *source node* (subject) connects to *destination node* (object) through an *edge* (predicate). The following set of triples represent a graph:

Yearly_Chocolate_Sales as triples

Subject	Predicate	Object
ACVM Company Sales Data	is	Yearly_KitKat_Sales
ACVM CompanySales Data	is	Yearly_FruitAndNuts_Sales
Yearly_KitKat_Sales	equals	160000
Yearly_FruitsandNuts _Sales	equals	200000
...
...
...

8.2.1.1 Graph Database as Collection of Triples

A collection of triples creates a semantic database, the two features of this database are as follows:

1. Can add additional properties, relationship, association or attribute with each triple.
2. Can include new entities and relationships, just as a tabular database adds additional rows, or a columnar-family database adds additional columns.

8.2.2 Resource Description Framework (RDF) for Graph Databases

A triple *instance identifier–property name–property value* uses a Universal Resource Indicator (URI) for instance identifier. Triples form a specialized graph database. A triple-store is also represented in RDF (Resource Definition Framework).

RDF is a simple, yet very effective language for representing information using triples. RDF is a W3C (World Wide Consortium) standard for storing a graph database. A graph database is thus a triplestore, which uses RDF. The features in RDF are as follows:

1. An RDF data file is similar to three columns of triples: subject-predicate-objects and are also similar to triplets of document-key-values in the MongoDB.
2. A standard RDF schema provides definitions of classes and relationships between the properties and classes; an RDF does not depend on a schema and is thus flexible.
3. Triples represent the nodes and edges; format of triple is ‘instance-identifier’, ‘property-name’ and

'property-value'; and format of identifier is URI. An instance identifier is like an entity in a SQL database, property name is like a *key*, and property value is like a *value* in a field.

4. RDF provides for inclusion of new entities and relationships, just as a tabular database provides for inclusion of additional rows, or a columnar-family database for additional columns.
5. RDF provides for inclusion of additional properties to the relationship, association and attribute in a triple.
6. Simple concatenation combines multiple datasets. The combined datasets are then used as a whole.
7. Splitting the triples into multiple lines does not change the collective meaning; therefore, sharding in data collections is easy.

The RDF software parses the lines in data file when processing the queries, analyzing, visualizing, reporting or any other operation. RDFLib is a Python library, which enables working with RDF. Number of contributors enriches the Python RDFLib continuously. The library contains (i) an RDF/XML parser/serialize and (ii) in-memory and persistent graph backend.

The following example explains the entries at a graph database:

EXAMPLE 8.3

Recapitulate Figure 8.2. It showed a snippet of graph database of ACVM Company daily and yearly sales. Write the triples in RDF.

SOLUTION

The following are steps in the RDF lines in a data file:

Step 1: Write two lines to specify the URIs for resource location.

```
@prefix acvmCompanyData: <http://acvncmpany.org/data/>
@prefix salesFigure: <http://acvncmpany.org/salesRecord/>
```

Step 2: Write triples for ACVM of ID, acvm8268. They are as follows:

```
acvmCompanyData:acvm8268 salesFigure: Yearly_KitKat_Sales "160000"
AcvmCompanyData: acvm8268 salesFigure: Yearly_FruitAndNuts_Sales
"200000"
```

8.2.3 SPARQL Querying Language for RDF Graph-Database

Spark Query Language (SPARQL) is a query language for RDF graph database. SPARQL is W3C accepted query language. Features of SPARQL are as follows:

1. Allows taking the data without definition for separate schema and considers a schema as part of the data itself (Schema information may be provided separately, which enable joining of datasets without any problem)
2. Provides query operators needed during graph analytics
3. Provides JOIN, SORT, AGGREGATE operators
4. Provides syntax for specific graph path traversals
5. Includes queries for conjunctions, disjunctions, triple patterns and optional patterns in triples

6. Provides for querying data using graph traversal along a path. Traversal may be single step, path expression, or full recursion. RDF (Resource Description Framework) is a specialized query language.

[Path means a finite or infinite sequence of edges in a graph that connect a sequence of vertices. Graph traversal uses a path along the connected edges. Path expression means an expression consisting of path names using ORs. The expression denotes a set of paths between two start and end nodes. (Sign Plus or OR is used between the terms in an expression to express a set of path traversals.)

For example, consider a path expression, V1V3 +V1V2V3 +V1V2V2V3+ ... First term V1V3 means path starting from V1, directs to V3. The term V1V2V2V3 means V1 to V2, then V2 output back to input of V2, and then V2 to V3.

Path recursion means repeated traversal of path till a certain condition satisfies. The following example explains how to write a SPARQL query and the output after query processing.

EXAMPLE 8.4

Recapitulate Example 8.3. (i) Write a SPARQL query for output of RDF triples. (ii) What will be the result from the query processor?

SOLUTION

- (i) The following are the steps for query in SAPRQL:

Step 1: Specify the URIs for resource location.

```
PREFIX acvmCompanyData: <http://acvmcomapny.org/data/>
PREFIX SalesFigure :< http://acvmcomapny.org/salesRecord/>
```

Step 2: Write query.

```
SELECT ? property ? value
WHERE {acvmCompanyData: acvm8268salesFigure ? property ? value}
```

The ? mark is a wildcard as the query is for selecting all properties and their values for acvm8268 salesFigure. If instead of ? Yearly_KitKat_Sales is mentioned, then only sales figures for Kit Kat retrieve from the RDF data-file.

- (ii) Following will be the output from the query:

```
Yearly_KitKat_Sales "160000"
Yearly_FruitAndNuts_Sales "200000"
```

8.2.4 NativeDB Graph Database

Relational DB distributes the relationships and stores as tables. Path traversal from a vertex to vertices retrieves the multi-step relationships. An efficient storage mechanism with ease in path traversals is thus needed, especially in a Big Data environment, which consists of distributed data-clusters and parallels computing data-nodes.

Neo4j developed Native Data Store for graph databases.¹ NativeDB focuses on efficient use of available computing resources. The design is architecture aware design. NativeDB graph stores nodes and relationships directly. Direct storage makes retrievals efficient. Figure 8.3 shows the relationships in use during path traversal and in use during operations in NativeDB.

Features of native data store are as follows:

1. Design provides for workload of memory management, query engine, and query language at storage
2. Design provides the safe storage, efficient querying consistently and without the aid of other components
3. Organizes the graph data and models both graph structure, vertex properties and edge properties
4. Represents the graphs in-memory and on-disk
5. Caches the graph data in-memory either in batch mode or on-demand from the on-disk
6. Enables timestamps
7. Persisting updates of graph along with the timestamps from in-memory graph to on-disk

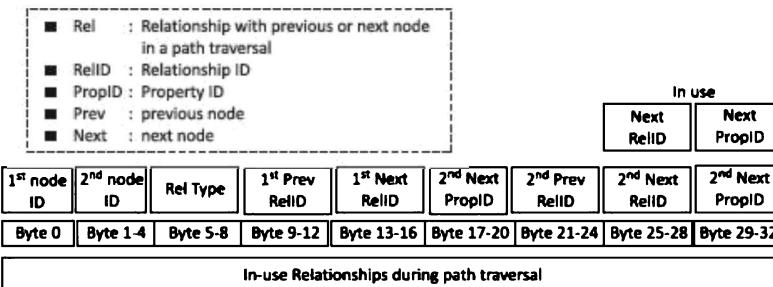


Figure 8.3 Native data graph relationships

8. Provides graph data streaming, graph data updates for modifying the graph structure and/or property data accordingly
9. Provides addition of the edges, removal of vertices and updates of properties
10. Performs querying of graph data by loading the graph structure and/or property data
11. Finds neighbours of a vertex, retrieves property of an edge by path traversals.

Neo4j developed a query language called *Cypher* for the NativeDBs. Tests for execution times in the searches for 2nd, 3rd, 4th and remote neighbours show that the Neo4j Native Data Store is faster compared to other formats. IBM G system² for graph analytics, visualization and other graph applications also support NativeDBs.

8.2.5 Property Graph Model

A property graph assigns property to the nodes and edges. The following example explains a property graph:

EXAMPLE 8.5

- (i) How do you draw a property graph for a student data? Student data contains student Id, semester, subject options, grades and teacher field.
- (ii) How do property graphs model the RDBMS tables?

SOLUTION

- (i) Figure 8.4 shows the property graph of student data with StudentId, semester, subject options, grades and teachers:
- (ii) Figure 8.5 shows RDBMS tables to which the above property graph model corresponds.

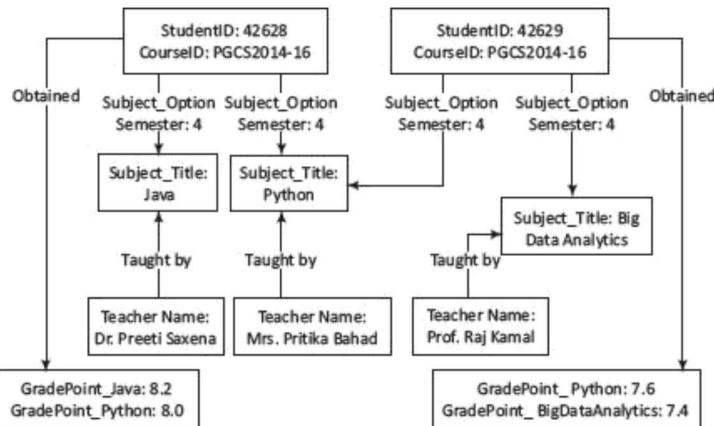
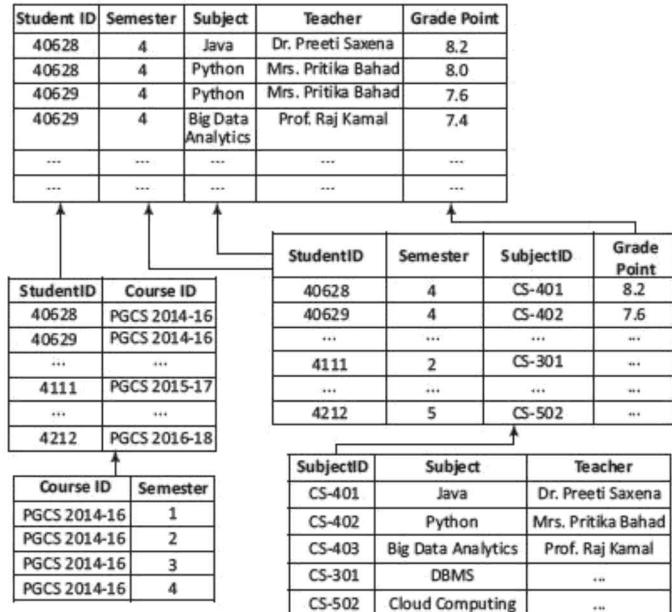


Figure 8.4 Property graph of students, semester, subject options, grades, and teachers

Figure 8.5 RDBMS tables for



students, semester, subject options, grades and teachers

Three tables in the figure store the values separately. Queries in the program search and select the related column value. An arrow in the figure shows the link of a column in table with the next related table. Property graph model (Figure 8.4) clearly shows the relationships and associations as compared to RDBMS tables (Figure 8.5). Property graph model represents the relationships or associations pictorially and are thus easy to interpret.

Self-Assessment Exercise linked to LO 8.1

1. How does Data Store model as a graph? How is a node represented? How is an edge represented?
2. How do the nodes and edges represent the properties?
3. How does a graph depict relationships? Give two examples.
4. How does a graph represent knowledge and reasoning? Give two examples of each.
5. How does a graph represent parameters? Give two examples.
6. How does a DFG represent the flows of data and program.
7. List the meaning of the terms: (i) triple ‘subject-predicate-object’ format, (ii) triplestore, (iii) instance identifier–property name–property value RDFformat, and (iv) NativeDB.
8. List characteristic features of the Resource Description Framework.
9. How does a property graph represent RDBMS tables?

8.3 GRAPHS, NETWORK ORGANIZATION AND GRAPH ANALYTICS

A network organization means where the persons or entities interconnect with others and have areas of common interest, business or study. A model of graph data store is a network organization. Graph network examples are weblinks network, social network, business network and students network. A network graph consists of vertices V , interconnected by directed or undirected edges E .

LO 8.2

Graphs, graph network organization, choosing graph for analytics and graph analytics use cases

A network consists of items (entities, persons or web page links) and relationships or the associations between items. The graphs, such as trees, can store a relational DB. However, relational DBs have the following problems:

1. Mostly cumbersome to navigate
2. Difficult to scale up
3. Difficulties in adding new relationships or associations.

Graph Data Store nodes are easy to navigate through a traversal of paths. Path traversals can be optimized for faster traversing. Scaling up and adding new relationships or associations are easy in the Graph Data Stores.

8.3.1 Network Organization

Graph model of a network organization helps in following tasks:

1. Detecting patterns
2. Finding organization inherent in the network

3. Finding communities and micro-communities. For example, finding the groups where new trends are emerging, and the groups of similar interests
4. Finding connections. For example, finding the student-specific connected groups showing preference for programming language subjects
5. Modeling communication. For example, finding the specific events which create active interest in the specific community-groups, such as launching a new luxury car model which creates interest in managers and chief executive officers, or starting a new course on Python which creates deep interest in postgraduate Computer-Science students
6. Modeling collaborations. For example, finding the collaborating and sharing communities which have similar interests, finding community sharing new techniques and algorithms that has interest in Big Data Analytics
7. Modeling influences. For example, finding the persons (nodes) with high degrees of interconnections to large number of entities compared to the others
8. Modeling distances between set of entities. For example, finding how much distance a path-traversal takes place on an average when searching a common-interest entity
9. Making discoveries and providing the previously unknown information after the search and analysis

Distance in a graph consisting (V and E) refers to the number of edges connecting the two vertices v1 and v2 on the path traversal between them.

8.3.2 Probabilistic Graphical Network Organizations—Bayesian and Markov Networks

Probability means the chance of observing a dependent variable value with respect to some independent variable. Suppose a Grandmaster in chess won 22 out of 100 matches, 78 matches were a draw, and lost none of the matches. Then, the probability P of winning P_w is 0.22, P of drawn game P_D is 0.78 and P_L of losing,

$P_L = 0$. The sum of the probabilities normalizes to 1, as only one of the three possibilities exist.

Probability states mean that states of P values as a function of all possible independent values, situations or variables. For example, if probabilities of winning, drawing and loosing for Chess-player supercomputer AlphaZero against a Grandmaster are $P = (0.22, 0.78, 0)$, then the probability function has three states. Here, the probability state is a discrete function. Sum of P values is one.

Probability distribution means that distribution of P values as a function of all possible independent values, variables, situations, distances or variables. For example, P is given by a function $P(x)$ and P varies as x changes. Variations in $P(x)$ with x can be discrete or continuous. Here, again values are normalized such that sum of the P values is 1 (Section 6.2.5).

The probabilities distribute in the entities. The vertices represent the entities. The probability distribution function $P(x)$ distributes at the neighbouring vertices of a parent. That means vertex x has a property with probability $P(x)$, where x is the distance from the parent vertex. Neighbouring means associated, influence or effected vertices of the parent.

A *Bayesian Network Graph* (BNG) is a graph where each node represents a random variable in a DAG. The

variable has a probabilistic distribution over the connected nodes. No cyclic path traversals occur in BNG during querying or computations.

Markov Network Graph (MNG) is a Graph where each node represents a random variable in an undirected graph. The variable has a probabilistic distribution over the connected nodes during path traversal. The cyclic path traversals may also take place in an MNG.

Propagation of a property in a network implements by a function, $\text{func}(x)$ where x refers to neighbourhoods of the associated vertices ($\text{vertex}^{1^{\text{st}}}, 2^{\text{nd}}, 3^{\text{rd}}, 4^{\text{th}}$ and so on) with respect to a parent vertex. The $\text{func}()$ is a propagation function which represents an inference, evidence, belief, expectation or influence as a function of distances. The distribution (propagation) over distance can be continuous or discrete (Section 3.2.1 for definition of distance between the vertices).

$\text{Func}()$ can be a user defined function (UDF), vectorized UDF (VDUF) or group vectorized UDF (GVUDF) [Sections 5.3.2.2 to 5.3.2.4]. $\text{Func}()$ can be a probability distribution (propagation) function $P(x)$ or joint conditions probability function or potential function.

Probabilistic graphical models, such as Bayesian networks or Markov networks have number of applications. Probabilistic graphical models are used in machine learning. The models provide a compact representation. The model, for example Bayesian network, provides solutions for probabilistic inferences. Examples of applications are business strategies, risk assessments, medical diagnosis, speech recognition, assessment of loan risk default, etc.

Bayesian Network Graph (BNG)

A BNG network organization has the following features:

1. It is a directed graph model in which non-cyclic and no reverse path traversals take place for computations
2. Enables a compact representation which gives probabilistic relationships among a set of variables
3. Enables the computations of joint probability distributions over the probability state variables
4. Each node has a set of conditional probabilities which specifies quantitatively the influences (effects) of the parent
5. The property values at vertices have Conditional Probability-Distribution (CPD), table of graph nodes, node properties and probabilities, called Condition Probabilities Table (CPT)
6. An edge between two nodes means that these two nodes have conditional probabilistic dependency. A missing edge between two nodes means conditional independence of the node from the parent node.

The following example explains the concept of CPD and CPT.

EXAMPLE 8.6

Assume that a student chooses subjects in a semester examination and obtains GPAs and the probabilities of obtaining GPAs in the subjects are as follows:

Subject	Graph Node	Probability P_{8+} for GPA 8.0 or above	P_{8-} for GPA below 8.0
Subject 1	V1	0.85	0.15
Subject 2	V2	0.8	0.2
Subject 2 theory	V3	0.75	0.25
Subject 2 practical	V4	0.80	0.2
---	--	---	---
---	--	---	---

Number of rows in the table above depends on the subjects and their nature (theory, practical or general).

V1 connects V2 by a directed edge e1 towards V2. V2 connects V3 by a directed edge e2 towards V3 and also connects to V4 by a directed edge e3 towards V4. Using this graph:

- (i) Show diagrammatically Bayesian-Network Graph for obtained probabilities of GPAs with probability using distribution of probabilities P in different subjects, and
- (ii) Explain the concept of conditional probability distribution (CPD) and conditional probability table (CPT).

SOLUTION

- (i) Figure 8.6 shows Bayesian network graph for a student obtaining GPAs with probabilities distributions in the GPAs:

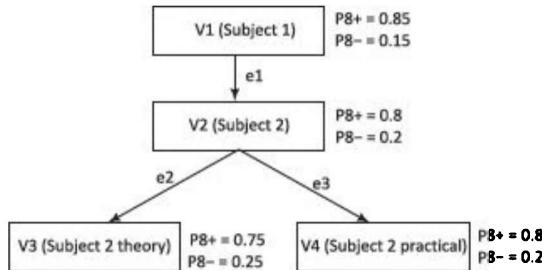


Figure 8.6 Bayesian network graph for a student obtaining GPAs with probabilistic distributions in the GPAs

- (ii) Following are the probabilities distributions at the nodes:

V1	$P(V1)$	V2	$P(V2)$	V3	$P(V3)$	V4	$P(V4)$
8-	$p_{10} = 0.85$	8-	$p_{20} = 0.8$	8-	$p_{30} = 0.85$	8-	$p_{40} = 0.8$
8+	$p_{11} = 0.15$	8+	$p_{21} = 0.2$	8+	$p_{31} = 0.15$	8+	$p_{41} = 0.2$

The first column shows the property at V1, 8- means the student obtains below 8.0 and 8+ means above 8.0. The probability distribution values enable the computation of conditional probability, cp values and give the CPTs. The formats of CPTs are as follows:

V1 V2 P (V2 V1)	V2 V3 P (V3 V2)	V2 V4 P (V4 V2)	V3 V1 P (V3 V1)
8- 8- cp ₂₁₀₀	8- 8- cp ₃₂₀₀	8- 8- cp ₄₂₀₀	8- 8- cp ₃₁₀₀
8- 8+ cp ₂₁₀₁	8- 8+ cp ₃₂₀₁	8- 8+ cp ₄₂₀₁	8- 8+ cp ₃₁₀₁
8+ 8- cp ₂₁₁₀	8+ 8- cp ₃₂₁₀	8+ 8- cp ₄₂₁₀	8+ 8- cp ₃₁₁₀
8+ 8+ cp ₂₁₁₁	8+ 8+ cp ₃₂₁₁	8+ 8+ cp ₄₂₁₁	8+ 8+ cp ₃₁₁₁

cp₂₁₀₀ is probability when the conditions at nodes V1 and V2 correspond to GPs below 8.0 in both subjects. Similarly, other cp values are probabilities when conditions given at rows 1 and 2 are true. The CPTs enable computations of the probability of the conditions such as V1 = 8+, V2 = 8+, V3 = 8+, V4 = 8+ being true. Four nodes have $2^4 = 16$ conditional probabilities in the Bayesian network. 16 conditions are possible for four nodes of the graph.

Markov Network Graph (MNG)

A Markov Network Graph (MNG) is a network organization which is undirected and can have cycles in path traversals.

Assume that all vertices are reachable from a starting vertex. Breadth first traversal (search) [BFS] is used when the graph has cycles. Therefore, the visited vertices are marked. The marks at each visited vertex can be stored in an array of bits (Booleans).

8.3.3 Graph Analytics

Three types of processes for graph analytics are (i) performing searches and finding matches, (ii) performing topological analysis (for example, analyzing degree and degree distribution, closeness, betweenness, centrality parameters), and (iii) traversing the path and studying the flow, [for example, probability flow (variation with respect to distance)].

APIs and tools of graph analytics do the following:

- (i) Find association of nodes, which implies finding a node relation or connection with the next or previous node
- (ii) Detect patterns, communities and micro-communities
- (iii) Do a graph search and find graph matches for specific connected groups which show similar preferences
- (iv) Find collaborations among entities having similar interests (for example, collaborations by sharing similar techniques and algorithms)
- (v) Do shortest path analysis for communications
- (vi) Analyze the distances between a set of common interest entities
- (vii) Compute the centralities (degree centrality, closeness centrality, betweenness centrality, Eigen vector for centralities of the neighbours to a node of higher centrality than the neighbours)
- (viii) Detect the anomaly and discover previously unknown information
- (ix) Evaluate the influence distribution by comparing the nodes of high degrees of interconnections with large number of entities with respect to others,

(x) Find the ranking from PageRank, a term used in analogy with web PageRank.

Graph analytics use the methods of collaborative filtering (Section 6.4.3), stochastic gradient method (Section 6.7.3), triangle counting (Section 9.5.6), K-core analysis (Section 9.5.3), Web communities (Section 9.4.6), social communities (Section 9.5.8), and PageRank (Sections 9.4.1 and 9.4.3).

Node distance refers to the number of edges connecting a node from a node taken as origin. If two vertices v₁ and v₂ on path traversal are the nearest neighbour, then distance of v₂ from v₁ is 1; if the next nearest neighbour then 2; if next to next then 3, and so on.

A node distance is a measure of number of 1st, 2nd, 3rd, 4th and so on and corresponds to the neighbourhood associated with a vertex.

The parameters nn₁, nn₂, measure the node neighbourhoods, where nn₁, nn₂, ... = number of 1st neighbour node, 2nd neighbour node, ..., respectively, and so on to a querying node.

The *node centrality* of a node is defined in reference to other nodes using the metrics. Metrics for centrality are degree, closeness, betweenness or other characteristic of the node, such as rank, belief, expectation, evidence, reputation or status.

Nodes *closeness* to a vertex u is defined, as reference to other connected vertices in V_u with u. V_u is a subset of vertices in V that connect with u. The centrality (closeness index), c_C is function of distances of vertices.

$$c_C(v) = \sum_{u \in V} [d(u, v)]^{-1}. \quad \dots (8.1)$$

where d (u, v) is the distance between u and v when traversing the path, and u and v are elements of V_u. Summation is overall connected vertices to u [Closeness is inverse of the node distance in terms of neighbourhood 1, 2, ... or N (Recall K-NN in Section 6.3.6)].

Node *betweenness* defines the extent to which a vertex is located 'between' other pairs of vertices, measured by the number of times a node is present between the shortest paths. Betweenness c_B (v) relates to the number of pairing nodes.

Betweenness centrality of a vertex v requires calculating the lengths of shortest paths among all pairs (p, q) of vertices connected to u, and computations of summation for each pairing vertex in V_u. The centrality (betweenness index), c_B (u) is function of shortest distances with the pair of vertices.

$$c_B(u) = \left\{ \sum_{p \neq q \neq u \in V} \alpha(p, q | u) \right\} \times [\alpha(p, q)]^{-1}. \quad \dots (8.2)$$

$\alpha(p, q | u)$ is the shortest distance between u and pair (p, q). Divisor of the sum, $\alpha(p, q)$ is the shortest distance between p and q. $\alpha(p, q)$ is the normalization factor. Summation is overall the connected pairs of vertices to vertex u. Vertex u and vertex pairs (p, q) are elements associated with V_u subset of vertices.

EXAMPLE 8.7

Figure 8.7 shows a multi-directed graph with 12 vertices (p, q, r, s, t, u, p', q', r', s', t' and u').

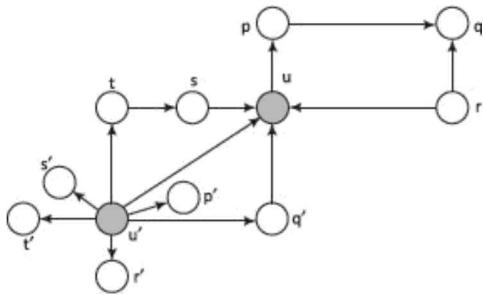


Figure 8.7 Graph with 12 vertices and 13 edges

- Compute order of the graph, in-degrees and out-degrees of u and u' , and 1st, 2nd, 3rd neighbourhoods of u .
- Compute the centralities, closeness and betweenness using Eqs (8.1) and (8.2).

SOLUTION

- Order of the graph is given by number of vertices $N_v = 12$ and edges $N_e = 13$.

In-degrees of $u = 4$. Out-degrees of $u = 1$. In-degrees of $u' = 0$. Out-degrees of $u' = 7$

1stneighbourhood of $u = 5$ (p, q', r, s, u').

2ndneighbourhood of $u = 7$ (p', q, r', s', t, t', u'). [u connects by path traversal through two edges.]

3rdneighbourhoods of $u = 8$ ($p, p', r, r', s', t, t', u'$). [u connects by path traversal through three edges.]

- Centrality closeness of u , $CC(u)$ computes as follows:

Distances with the five nearest neighbours, p, q', r, s and $u' = 1$ each. Distances with the seven next to nearest neighbours p', q, r', s', t, t' and $u' = 2$ each. Therefore, $C_C(u) = (1/5 + 2/7) = 0.48$ using equation (8.1).

Centrality betweenness index C_B of u computes as follows:

Connected Number of vertices pairs = 8 [(u', t'), (u', s'), (u', r'), (q, r), (p, q), (s, t), (q', u'), (u', p')]

Shortest distances with eight nearest pairs = 1 each. No other pairs connect u with distance = 2. Therefore, $C_B = 8$.

8.3.4 Choosing Graph Analytics

When does graph analytics help? Loshin³ in a study suggests that graph analytics can help in business or other problems where the following characteristics require analysis:

1. Connectivity from the number of relationships and association types
2. Undirected unidentified patterns, undirected graph path traversal, discovering new unidentified pattern (for example, finding students of new batches opting for Natural Language Processing

- subject, or finding emerging pattern for cycling near the sea beach)
3. Missing pattern (for example, finding none of computer science student interested in financial analysis in an organization)
 4. Discovering new knowledge [for example, evolving interest in Big Data Analytics course in students of UG Computer Science (Example 8.5)]
 5. Analysis using the ad hoc queries for finding reasons
 6. Predicting interactive performance, which means predicting interactions among entities on events.

8.3.5 Use Cases of Graph Analytics

Following are use cases of graph analytics from different domains including the business domain:

1. Monitoring and analysis of social media
2. Analysis of an enterprise social network: The analytics enables search of expertise, knowledge, recommending expertise, experts location, and detecting of spam and anomaly
3. Financial analysis for undertaking the financial decisions, new knowledge discovery, security analysis, anomaly and fraud detection, such as credit card frauds and fake bill payments, account manipulations
4. Commerce and trade analysis for customer behaviour prediction, planning and strategies for sales promotion, price discovery, detecting weak supply chain links and commerce frauds
5. Cellular network analytics in telecom companies for increasing their operations or helping police to track criminals
6. Disease diagnostics, patient and disease analytics in healthcare studies, health care quality analysis, medicine research and development in genomics
7. Analyzes breaches in cyber security, location of breaches, detects denial of service (DoS) attack, and finds the attack sources.

Self-Assessment Exercise linked to LO 8.2

1. How does a network organization model the communications, collaborations and correlations using the graph?
2. How do the centrality parameters specify the degree, closeness and betweenness? Demonstrate using examples.
3. How does propagation of probability distribution function over the vertices enable decisions? When does the Bayesian network graph model the distribution of influence?
4. When is graph analytics used?

8.4 GRAPH ANALYTICS ALGORITHMS AND APPROACHES

A graph model for data store provides additional information about associations. Graph analytics thus enables analysis of additional properties, besides the ones using standard RDBMS or data warehouse framework.

Following are examples of algorithms for graph analytics which relate associations of entities:

Graph parameters, methods, diagnostics, decisions, statistical model, StatsModel, and probabilities-based analytics, and technical complexities in analyzing the graphs

8.4.1 StatsModel and Probability Based Analytics

Centralities Parameters in Graphs

Computations of the centralities of entities in terms of (i) in-degree in nodes of directed graphs and (ii) out-degrees, (iii) distribution of degrees among the nodes, (iv) closeness, (v) betweenness, (vi) other parameters such as effective closeness, reputation, status and link rank. Degree of a node is the number of edges linked to a node. Node closeness to other vertices depends on the sum of distances with other vertices. Node betweenness relates to the number of pairing nodes. Node relations or connections are associated with the next node and with the previous node.

Computation of financial worth of a company uses three parameters of centralities. Another application is monitoring the attacks on the network.

Path and Flow Analysis of Graphs

Path and flow analysis algorithms analyze the shapes (triangles, hexagons, trees and junction trees), shortest paths and top-K shortest paths. Algorithm for triangles counting finds the number of triangular relationships among the nodes. Top-K shortest paths means finding distances of the multiple paths which connect the vertices and which have the shortest paths among top K. Value of K is 2, 3, 4 and so on for top-2, top-3, top-4 and so on.

Matching and Search Analytics in Graphs

An algorithm matches the graphs and subgraphs after a graph search on path traversals. A filter algorithm uses the label, vertex-property, edge-property or geographical location for filtering the graph vertices. Collaborative filtering algorithm also does the searches. Collaborative filtering means finding matches in a bipartite weighted graph.

A subgraph is graph whose vertices and edges are subset of another graph. A graph $G1 = (V', E')$ is a subgraph of graph $G = (V, E)$ if $V' \subseteq V$, and $E' \subseteq E \wedge ((v_1, v_2) \in E' \rightarrow v_1, v_2 \in V')$ in set theory notations.

[Set theory uses symbols as follows: Symbol (i) \subseteq is for ‘subset of’, for example V' is subset of V . (ii) \wedge is logic symbol for ‘and’. (iii) \in is for ‘element of’ (for example $v_1, v_2 \in V'$ means v_1 and v_2 are elements of V').]

Collaborative Filtering (CF)

Collaborative Filtering (CF) is a technique used by recommender systems (Section 6.4.3). The system collects references or test information from many collaborating users. The system makes predictions about the interests of a user and then recommends to new users.

Detection of Clusters

Clustering analysis identifies the groups of special cases. For example, in a study of effect of discount

offered in versus sales, the identification of clusters enables the price discovery. Another example, a system identifies a cluster of students with deep interest in Big Data analytics. This enables a department to start additional new course in that area.

Detection and Analysis of Patterns

Pattern detection and pattern analysis are required in many applications. For example, identifying patterns of sales increase after an advertisement of a car model. That helps a car company for planning future advertisement strategies. Graphical detection of patterns identifies new patterns, which may lead to new opportunities (for example in education, business or health care)

Anomaly Detection

Anomaly detection and analysis find the abnormal behaviour, structure, feature, content or semantic features. Anomaly detection may also help in its usability and summarising anomaly attributes. It enables identification of spam source and can lead to detect frauds related to credit card, medical claim or detecting fictitious transactions.

Community and Network Analysis

The community and network analysis analyses the close-by entities, and fully mesh-like connected sets. The network graph analysis besides the centralities analysis, also find *page rank* of the links.

Community and network analyses use triangle count, clustering coefficient, K-neighbourhood, connected component and K-core analysis parameters.

K-neighbourhood analysis means finding the number of 1st neighbour nodes, 2nd neighbour nodes and so on. (K = 1, 2, 3, 4 and so on).

K-core analysis means number of cores within a marked area. A core may consist of a triangle of connected vertices. A core may consist of a rectangle with interconnected edges and diagonals. A core may also be a group of cores.

PageRank Based Analytics

PageRank is a metric for importance of each vertex in a graph. Assume an edge from v1 to v2 represents endorsement of importance of v2 by v1 by a connection. The importance of v1 results from the interactions between them by creating a relationship between them, sharing a belief or some other mean (Term PageRank borrows from web PageRank).

StatsModel and Probability Based Analytics

Statistical model is a class of mathematical models. It considers a set of assumptions for the sample data representing a larger population. A statistical model represents, often in considerably idealized form, the data generating process.

StatsModel refers to a Python module that provides classes and functions for estimation of many different statistical models, as well as for conducting statistical-tests, and exploring the statistical data.

Analysis of Big Data need reasoning and prediction under uncertainties. Uncertainty requires usages of probabilities and StatsModel. Many critical Big-Data graph analytics uses the inter-relationships between the entities. Graph based representations need learning of graph structure, computations of conditional independence, and probabilistic inference.

Diagnosis and Decisions based on Probabilistic Graphical Models

Bayesian networks or Markov networks are used in various applications, such as assessment of loan default risk. Graph analytics algorithm first convert the network into junction trees and then, performs the graph traversal. This is due to use of distributed file system in Big Data graph database. A network probabilistic graph-model needs intensive numeric operations.

Advanced Concepts in Probabilistic Graphical Models

Use of Junction Trees Graph (JTG) formed for the Bayesian Consider four nodes BNG with 2 states per node (Example 8.6). That needs $2^4 = 16$ CPT entries for four probable conditions at all four vertices when jointly considered.

Assume a fifteen node BNG with five states per node. Consider five states at each node in Example 8.6. Assume that five states of GPA awarded probabilities are P8+, P8-, P6+, P6- and P4-. That needs 5^{15} CPT table entries. Therefore, 5^{15} CPT entries for five probable conditions at all 15 vertices when jointly considered. Therefore, an algorithm first reduces the Bayesian network graph to a JTG using a junction tree algorithm (JTAl).⁴

The number of incoming directed edges connect at each junction. A junction tree is a one that has a root node at a junction, which has number of directed edges to a set of junctions (daughter nodes). Each daughter is again a junction, which has number of directed edges to a next set of junctions. Thus, a tree-like structure exists starting from the root.

JTAl is a machine learning algorithm. JTAl is also called clique tree method. The method extracts *marginalization* in general graphs. It propagates *belief* (evidence collected at the junction from number of connected nodes in the junction tree, also called *inference* from that tree). JTAl does the belief computations for each junction tree. Usage of JTG eliminates the cycles in traversals also.

Probabilistic inferences at junction trees Probabilistic representation of the distributions in terms of the CPTs is replaced by *inference* computed for each junction tree of JTG. An algorithm computes Potential Tables (POTs) which replace the use of CPTs for analyses.

Knowledge Discovery

Algorithms for graph analytics along with machine learning algorithms lead to new facts. They discover new associations during the analyses. They lead to discovery of new knowledge.

8.4.2 Technical Complexity in Analyzing Graphs

Big Data analytics has challenges due to need of compact representation and parallelism of the computations when large datasets execute on the clusters. Following are the technical complexities in the Graph analytics:

Graph-Partitioning Complexity

Triangularly connected nodes or similar structures of high connectivity may be of specific interests. Big Data analytics depend on the distribution of data in HDFS-like files or distributed DBs. High connectivity structures pose the problem of partitioning and data sharding. When the data structures have such problems, the network access intervals increase significantly performance of the algorithm also reduces.

Graph-memory Accesses Unpredictability

Big Data analytics need parallelization of computations. Several path traversals thus initiate at the same instance. However, due to sequential node links through the edges, each path traversal follows a

sequence of nodes.

Many traversals are necessary when finding new patterns. Poor locality and irregular memory accesses are the problems. The memory access times are unpredictable for each paralleled traversal. Much time is thus lost in accessing memory.

Analytics Problems and Topologies

Graph characteristics may inhibit the need of scalability due to performance issues. The large graphs and number of topological structures result in complexities.

Dynamic Interactions for Graphs

The dynamic computations enable update of cluster and subgraph. Graphs in a Big Data application need dynamic analysis of data with 3Vs characteristic. Therefore, rapid response issues are complex in case of dynamic interactions. Computations in dynamic graphs have poor locality and unpredictable access times from memory. The workload is also hybrid (sequential and parallel).

Self-Assessment Exercise linked to LO 8.3

1. How do the additional information about associations enable new parameters from analytics besides the one using standard RDBMS?
2. What are the graph parameters derived from graph analytics? Write equations for each of them.
3. Recapitulate Figure 6.9. How do the cluster results from the graph databases provide additional information compared to results shown in the figure?
4. How does graph-partitioning problem result in technical complexity during graph analytics?

8.5 | SPARK GRAPHX PLATFORM

IBM System G offers a set of Big Data tools for graph computations. G (stands for graph, which may be a property graph, Bayesian network graph, or cognitive network graph. A graph may be static or dynamic, small or large, topological or semantic. G has library functions for graph analytics. G applications include creating and analyzing the database, visualization and middleware for graph.

LO 8.4

Apache Spark GraphX, its features, architecture and components, and their applications for graph-analytics

Apache Spark GraphX is open source software that provisions a number of functions and operators for graph stored in HDFS environment. Apache Spark refers to a multi-component platform for Big Data computing that uses data store at a HDFS file system, HDFS compatible data sources, such as HBase, Cassandra, Ceph or S3. Spark enables the use of data frames in *Resilient Distributed Datasets (RDD)* (Section 5.4.2), and the creation of ETL pipelines (Section 5.5). An RDD is a collection of objects distributed on many computing nodes. Spark standard APIs enable creation of application APIs in Scala, Java, R and Python (Chapter 5).

Graph analytics need functions which compute degree centralities, degree distribution, separation of degrees, betweenness centralities, closeness centralities, neighbourhoods, strongly connected

components, PageRank, shortest path, Breadth First Search (BFS), minimum spanning tree (forest), spectral clustering and cluster coefficient.

The following subsections describe Spark GraphX Architecture, fundamental operators, algorithms and their applications for graph analytics.

8.5.1 Features of a Graph Analytics Platform—Apache GraphX

Apache Spark provides a software component known as GraphX. GraphX 2.2.1 released on December 01 2017 is a new version. GraphX is an open source tool for graphs and parallel computations for graphs. GraphX tool processes a Resilient Distributed Graph (RDG). GraphX tool is a stack component of BDAS (Section 1.6.4.3).

GraphX provides a set of fundamental operators such as `subgraph`, `joinVertices` and `aggregateMessages`. Apache GraphX provides for computations using the property graphs. Identifier in GraphX is 16-bit long unique-key. Edges have vertexIDs for corresponding source-destination paths.

GraphX programming features are:

1. Includes a growing collection of graph algorithms and graph creator operators (builders)⁵
2. Extends the Spark for computing with Resilient Distributed Dataset (RDD) (Section 5.4.2). Graph builders do not repartition the edges by default, they are left in their default partitions (such as their original blocks in HDFS).
3. Introduces a new abstraction of graph called directed multigraph which has properties associated with each of the vertex and edge.
4. Provides several ways of building a graph; a graph builds from a collection of (V, E) [set of vertices and edges] in an RDD or on disk
5. Gives high performance and competes with the fastest graph systems
6. Retains features of Spark, ease in uses, flexibility, and fault tolerance
7. Unifies computations using iterative Graph, ETL, exploratory analysis within the GraphX because of association with Spark, PySpark, as well as StatsModel
8. View data, in the same manner as a graph and as a collection
9. Provides an optimized variant of the Pregel APIs
10. Enables writing of user defined iterative graph algorithms using the Pregel

Figure 8.8 shows GraphX architecture.

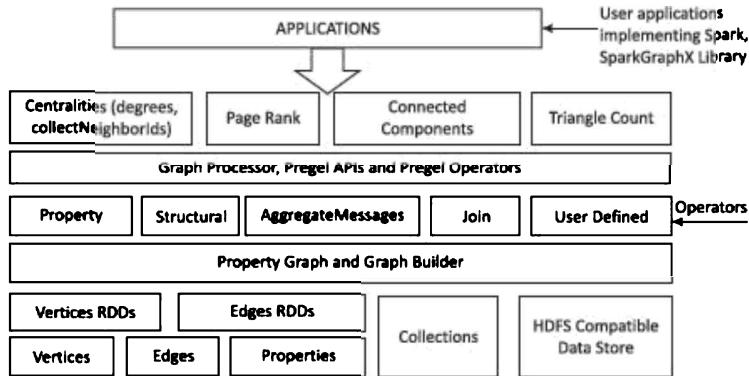


Figure 8.8 GraphX Architecture

GraphX operators

GraphX aggregation operator, page rank, connected components and triangle-counting algorithms do the following:

- Aggregation operator**—Several graph analysis tasks require aggregation of information about the neighbourhood of each vertex. Many iterative graph algorithms repeatedly aggregate properties of neighbouring vertices. Examples are computing the shortest path to a source, smallest reachable vertex id, connected components and PageRank. The operator applies a user defined `sendMsg` function to each *edge triplet* in the graph. Then, application uses the `mergeMsg` function, which aggregates those messages at their destination vertex.
- Pregel operator**—executes in a series of supersteps in which vertices receive the sum of their inbound messages from the previous superstep. (Superstep is a term used in Pregel vertex-centric processing in large distributed graphs. Supersteps do computations consisting of a sequence of iterations.) The operator computes fresh value for property at vertex. Then, that sends messages to the neighbouring vertices in the next superstep. GraphX Pregel-API provides computation of messages in parallel as a function of the edge triplet. A message for computation provide accesses to both attributes of source and destination vertices. When a vertex does not receive a message that is skipped within a superstep, the Pregel operator terminates the iteration and returns the final graph when no messages remain. Pregel operator in GraphX is a bulk-synchronous parallel-messaging abstraction that constrains to graph topology.
- PageRank algorithm**—Assume that an edge from v_1 to v_2 represents an endorsement of importance of v_2 by v_1 . The function measures the importance of each vertex in a graph. For example, if many students opt for a teacher's course, then, teacher's rank is high. If many followers follow a twitter account then that account rank is high. Similarly, a web page is searched by many then that page rank is high. GraphX computes PageRank statically as well as dynamically. Term PageRank is from Google web PageRanking system for searches (Refer Section 9.4.3 also).
- ConnectedComponents algorithm**—labels each connected component of the graph with the ID. Each connected component ID is an identifier of the lowest-numbered vertex. For example, in a social network, connected component objects can approximate clusters. GraphX contains an implementation of the algorithm for the `ConnectedComponentsObject`.

The function `graph.connectedComponents().vertices` computes the connected components of the datasets (for example, student-teacher datasets, and social network datasets).

1. **Degree Computation Objects:** Functions `graph.inDegrees.reduce(max); graph.outDegrees.reduce(max); graph.degrees.reduce(max)`; computes in-degree, out-degree and degrees in a graph. These functions analyze the degree distribution also at the vertices.
2. **Collection neighbour Ids and neighbours Operators:** The functions `collectNeighborIds(edgeDirection: EdgeDirection); collectNeighbors(edgeDirection: EdgeDirection)`
3. **Triangle Count Algorithm**—Determines the number of triangles passing through each vertex. The count is a measure of clustering. A vertex is part of a triangle when it has two adjacent vertices with an edge between them. TriangleCount **requires the edges to be in canonical orientation (`srcId < dstId`)** [Source vertex ID is `srcID` and Destination vertex ID is `dstID`]. Graph is partitioned using `Graph.partitionBy` operator. The function `graph.triangleCount().vertices` counts the triangles formed by vertices.

Super step (ss) is a set of steps performed. The following steps may take place in the framework during a superstep:

- (i) Receives and reads the messages that previous superstep $ss - 1$ had sent to v
- (ii) Applies a user-defined function $f_{udf}()$ to each vertex in parallel, therefore, $f_{udf}()$ essentially specifies the behaviour of single vertex v at a single superstep ss
- (iii) Can mutate the state of v
- (iv) Can send the messages to other vertices (for example, along outgoing edges) that the vertices will receive in the next superstep $ss + 1$.

Pregel approach is that all communications [such a variable increment, aggregation, or applying $f_{udf}()$] takes place between ss and $ss + 1$. The same $f_{udf}()$ applies within each ss to all the vertices in parallel.

Property Operators in Class Graph [VD, ED] are `mapVertices [VD2]()`, `mapEdges [ED2]()`, `mapTriplets [ED2]()`. These functions yield a new Graph.

Structural operators in Class Graph [VD, ED] use `subgraph()`, `mask()` and `groupEdges()`. Function `reverse()` returns a new graph with all directions reversed.

The function `graph.apply()` creates a graph from RDDs of vertices. Apply function arguments are `edges.vertices: RDD [(VertexId, VD)]`; `edges: RDD [Edge[ED]]`; and `defaultVertexAttr: VD = null`

Join Operator

Graph databases supports JOIN (Section 3.2). **Join Operator** joins the data from external collections (RDDs) with the graphs. Also, a function merges extra properties with an existing graph. Main join functions are `joinVertices` and `outerJoinVertices()`. The `joinVertices` joins the vertices with the input RDD and returns a new graph with the vertex properties obtained by applying the user.

Many examples are available at Github site⁶, which demonstrate uses of GraphX functions. These examples guide the development of codes for Big Data graph analytics.

Page Rank Analytics

GraphX provides static and dynamic implementations of PageRank as methods of the PageRank object:
ranksByUsername = users.join(ranks).map {case (id, (username, rank)) => (username, rank)}. A GraphX operation runs static PageRank for a fixed number of iterations.

The dynamic PageRank runs until the ranks converge. The run stops when the rank does not change by more than a specified tolerance. GraphOps enables calling these algorithms directly as methods on graph. The function graph.pageRank(0.0001).vertices does the following: When page ranking does not change during the run beyond the specified tolerance 0.0001 (1 in 10000) then the iterative process stops and the rank value converge.

The following explains the use of PageRank method on graph model for network organization of students modeled as a graph.

EXAMPLE 8.8

Consider a graph model for a network organization of students modeled as a graph. Some student's text-notes are widely exchanged with strongly connected ones compared to the others. Assume that set of students are in a file data/graphx/students.txt and a set of relationships between students is given in data/graphx/stronglyConnected.txt. What are the steps for PageRank computations for each student?

SOLUTION

- (i) Import GraphLoader using the program statement given below:

```
import org.apache.spark.graphx.GraphLoader
```

- (ii) Load the graph edges using program statement as follows:

```
val graph = GraphLoader.edgeListFile(sc, "data/graphx/stronglyConnected.txt")
```

Here, sc stands for Spark Context.

- (iii) Compute PageRank and get each student page rank from the text file using program statement as follows:

```
val ranks = graph.pageRank(0.0001).vertices
```

PageRank method runs using Pregel operator graph.pageRank().vertices and does bulk-synchronous parallel messaging over the vertices to compute the ranks of the vertices. (Message in present case is for computing the rank.)

The Pregel operator executes in a series of supersteps in which vertices receive the sum of their inbound messages from the previous superstep. Pregel computes using sequence of iterations till rank converges within specified tolerance.

- (iv) Find students from the text file using program statement as follows:

```
val Students = sc.textFile("data/graphx/students.txt").map {line=>
  val fields = line.split(",") (fields(0).toLong, fields(1))}
```

- (v) Join the ranks of the students and map them with the username and rank. Use program

statement as follows:

```
val ranksByUsername = students.join(ranks).map { case (id, (username, rank)) => (username, rank) }
```

- (vi) Print the results after making the strings separated by new lines using the program statement as follows:

8.5.2 Dedicated Appliances for Graph

Graph analytics can use the Berkeley Data Analytics Stack (BDAS) (Section 1.6.4.3), can use GraphX (Section 8.5.1), IBM System G², NativeDB¹, or the appliances dedicated for the graph-analytics.

A dedicated appliance is a computing platform for in-memory graph computing and native multi-threaded (MT) processing. In-memory graph computing means computing using data first taken into RAM memory from the data blocks which store the Big Data. Native MT processing means multiple program-threads processing at the platform in place processing at the cluster of data nodes. The appliance thus provides the faster IOs and results in very high performance in Big Data environment.

Dedicated appliances for graph are fully software based. They run graph analytics application on the existing server/cloud. They use triples based RDF format. They process using SPARQL query language.

Self-Assessment Exercise linked to LO 8.4

1. How do the Spark GraphX component enable Big Data analytics and high performance parallel computations?
2. How do property graph and directed multigraph provisioned in GraphX?
3. What are the uses of subgraph, joinVertices and aggregateMessages, PageRank, ConnectedComponents and Triangle Count algorithms?
4. Describe usages of Pregel API and operator for iterative computing on the RDG datasets.
5. How does GraphX unify iterative Graph computation, ETL, exploratory analysis within GraphX?

KEY CONCEPTS

aggregation

anomaly

association

Bayesian network

BDAS

belief propagation

betweenness

centrality

closeness

clustering
collaboration
connected component
connectivity model
correlation
CPT
degree distribution
Directed Acrylic Graph
directed graph
directed graph in-degree
directed graph out-degree
directed multigraph
distance
edge
evidence propagation
flow analysis
graph
graph analytics
graph database
graph edge
graph model
graph node
graph partitioning
graph traversal
GraphX
GUDF
GVUDF
IBM system G
influence
instance identifier
interaction modeling
join
junction tree
knowledge discovery

Markov network
native data store
NativeDB
neighbourhood
node
node association
node centrality
node degree
node distance
node neighbourhood
nodes betweenness
nodes closeness
order of graph
PageRank
path
path analysis
path expression
path recursion
path traversal
pattern
potential table
predictive analytics
Pregel API
probabilistic graph
probability
probability distribution
probability states
propagation function
Property Graph
property h
rank
RDBMS
RDF
RDFLib

RDG
relationship
risk assessment
semantic database
Spark
SPARQL
StatsModel
subgraph
triangle count
triple
Triplestore
UDF
URI
vertex
vertexID
VUDF
W3C standard



LO 8.1

1. Graph models the data stores and databases. A semantic database creates from a collection of triples. Triples format is similar to subject-predicate-object format in English sentences.
2. RDF is W3C standard triple format, *instance identifier-property name-property value*. A *URI* represents the identifier.
3. Two features of graph databases are as follows: A graph (i) can add additional properties with each triple relationship, association and attribute, and (ii) can include new entities and relationships, just as a tabular database adds additional rows, or columnar-family database adds more columns.
4. Nodes in graph are easy to navigate through traversing the paths using nodes (entities) or edges (relationships).
5. NativeDB provides fast execution as compared to other graphDBs.
6. Property graph model is used when relationships and properties also store in the database.

LO 8.2

1. A graph also models a network organization, such as social media and web links. A Graph models

the connections, communications, collaborations, influences, correlations and distances between the entities.

2. Probabilistic graphical models, such as Bayesian networks or Markov networks have several applications. Graph models a Bayesian network organization, which uses directed graph, non-reversal paths and cycles. Graph models a Markov network organization that have undirected graph, can have reversal paths and cycles.
3. Three types of graph analytics are (i) finding matches and performing searches, (ii) topological analysis, for example analyzing the centralities, degree, closeness and betweenness, and (iii) paths between vertices and flow of properties, such as probabilities.

LO 8.3

1. Graph analytics algorithms enable the determination of centralities of entities in terms of the (i) directed graphs in-degree, (ii) out-degree at the nodes, (iii) degrees distribution of the nodes, (iv) closeness, (v) betweenness, (vi) parameters such as effective closeness, reputation, status and link rank.
2. Graph analytics algorithms enable path analysis, detection of clusters, find anomaly, detection of patterns, communities and perform network analysis. Graph analytics algorithms enable diagnostics and decisions making from computations of conditional probability tables, junction trees graph, inferences, evidences or beliefs and potential tables.
3. Big data store in distributed database environment presents technical difficulties due to sequential path traversals resulting in graph partitioning problems, and the slower responses from memory during parallel execution of the queries.

LO 8.4

1. Apache GraphX is (i) software of Spark for graphs, (ii) creates and computes Resilient Distributed Graphs (RDG), and (iii) does parallel computations, which give the high performance, thus competes with the fastest graph analytics system. GraphX has features of Spark, such as ease of use, flexibility and fault tolerance.
2. Apache GraphX provides for creations and operations on property graphs. Identifier in GraphX is 16-bit long unique key. Edges have corresponding source and destination vertexIDs.
3. GraphX introduces new graph abstraction, directed multigraph, GraphX library of a set of fundamental operators, such as subgraph, joinVertices and aggregateMessages and inclusion of PageRank, ConnectedComponents, and TriangleCount algorithms.
4. GraphX unifies iterative graph computation, ETL and exploratory analysis using Pregel API and operator.

Objective Type Questions

Select one correct-answer option for each questions below:

- 8.1 A way of defining the centrality of a node in reference to other nodes, use (i) metrics. Metrics for centrality of a node are (ii) degree, (iii) closeness, (iv) betweenness, or other characteristics of the

node such as (v) rank, (vi) belief, (vii) influence, (viii) expectation, (ix) evidence, (x) reputation, and (xi) status of a node.

- (a) all except ii and vi
- (b) all vi and xi
- (c) all except ii, iii, iv and vi to x
- (d) all

8.2 (i) Each DAG node represents a variable, traverses in one direction only along the edges with no cyclic traversals. (ii) When each node in a directed graph has no cyclic traversals then the directed graph is also called acyclic graph, (iii) directed multigraph provisions multiple parallel edges and that enables multiple relationships between two entities, (iv) Bayesian network graph in each node represents a random variable in an undirected graph. The variable has probabilistic distribution over the connected nodes. Cyclic path traversals takes place in a BNG, and (v) each Markov network graph node represents a random variable in a DAG and that variable has probabilistic distribution over the connected nodes, and no cyclic path traversal takes place.

- (a) all except ii
- (b) i to iv
- (c) i to iii
- (d) all

8.3 (i) Graph model can represent a hierarchy, which the sequence of columns does not specify. Graph (ii) triple format is subject-predicate-object, (iii) triple format can be instance *identifier-property name-property value*, (iv) identifier uses a Universal Resource Indicator (URI) in RDF, (v) SPARQL is query language for (vi) RDF, and (vii) RDD. (viii) Graph database can add additional properties of each triple relationship, association and attribute.

- (a) all except vii
- (b) all except i and vii
- (c) i to vi
- (d) all except iv

8.4 (i) Property graph model does not show relationships and associations. (ii) Tables in RDBMS represent the relationships or associations, (iii) An RDF data file is similar to three columns in triples: subject, predicate and object, and (iv) also similar to document-key-value pairs in MongoDB. (v) RDF schema standard declares the classes and relationships between properties and classes, and (vi) RDF does not depend on a schema and is thus flexible.

- (a) all except i and ii
- (b) all except iii
- (c) all except iii to v
- (d) all except iv and vi

8.5 Graph model of a network organization can help in the followings tasks: (i) detect patterns, (ii)

finding inherent organization in the network, (iii) communities and micro-communities modeling, (iv) connectivity modeling, (v) collaboration modeling, (vi) influence modeling, (vii) belief propagation, (viii) distance modeling, (ix) closeness computations, and (x) betweenness computations.

- (a) all except vii and viii
- (b) all
- (c) all except vii
- (d) All except vi to viii

8.6 Use cases of graph analytics are (i) enterprise-network analysis (ii) social network analytics (iii) expertise search, (iv) knowledge recommendation, (v) expertise location, (vi) anomaly detection, (vii) Spam detection, (viii) financial analysis for financial decisions, (ix) health care quality analysis, and (x) detection of cyber security breaches.

- (a) all except vii
- (b) all except i
- (c) all
- (d) i to viii

8.7 (i) K-neighbourhood analysis means the number of 1st neighbour nodes, 2nd neighbour nodes, and so on. (K = 1, 2, 3, 4 and so on), (ii) The community and network analysis analyzes the (iii) close-by entities, (iv) fully mesh-like unconnected sets, (v) network graph analysis beside centralities, (vi) also does computations of the *property* of the links, (vii) rectangle counts, and (viii) clustering coefficient.

- (a) i to vi
- (b) all
- (c) ii to iv
- (d) i to iii, v, viii

8.8 (i) Apache Spark new stack component is GraphX 2.2.1. GraphX (ii) is a Linux based open source tool for graphs, (iii) does the graph parallel computations, (iv) creates and computes Redundant Distributed Graph (RDG), and (v) provides a set of fundamental operators such as *subgraph*, *joinVertices*, and *aggregateMessages*. GraphX (vi) does not have functions for property graphs, (vii) uses 64-bit long unique key as identifier, (viii) is a part of BDAS architecture, and (ix) have Edges and corresponding source and destination vertexIDs.

- (a) i, v to ix
- (b) all except ii, iv, vi, vii
- (c) all except ii and ix
- (d) all

8.9 (i) The Pregel operator in GraphX is a bulk synchronous parallel messaging abstraction, (ii) with no constraints to the topology of the graph, (iii) the Pregel operator executes in a series of supersteps

in which (iv) vertices receive the *sum* of their inbound messages from the (v) previous step. (vi) Superstep is a term used in Pregel edge-centric processing in large distributed graphs, and (vii) Pregel does computations consisting of a sequence of iterations.

- (a) iii, vi and vii
- (b) all except ii
- (c) all except ii and vi
- (d) all

8.10 ConnectedComponents algorithm in GraphX (i) labels each connected component of the graph with the ID, (ii) each connected component ID is ID of lowest numbered vertex. (iii) Connected component objects can approximate as clustering a social network, and (iv) GraphX contains an implementation of (v) ConnectedComponentsObject in the algorithm.

- (a) all
- (b) all except i
- (c) all except ii
- (d) i to iv

Review Questions

- 8.1 How are the characteristics of a graph parameterized using the centrality parameters? **(LO 8.1)**
- 8.2 What are the benefits of modeling Data Store as a graph? What are the benefits of graph databases compared to RDBMS? **(LO 8.1)**
- 8.3 List the differences between the RDF and NativeDB for graph Data Stores. **(LO 8.1)**
- 8.4 How does a graph model a network organization? **(LO 8.2)**
- 8.5 When does a graph model a Bayesian network organization? **(LO 8.2)**
- 8.6 Explain the uses of graph analytics for (i) finding matches and performing searches and (ii) topological analysis, for example analyzing centralities, degree, closeness and betweenness and (iii) path and flow analysis. **(LO 8.2)**
- 8.7 How are path analysis, clustering detection, anomaly and patterns detection, communities and network analysis used in applications? **(LO 8.3)**
- 8.8 What are the technical complexities in Big Data graph analytics? **(LO 8.3)**
- 8.9 How are conditional probability tables in Bayesian network organization graph computed? What are the limitations of using CPTs? How do Potential tables differ from CPTs? **(LO 8.3)**
- 8.10 How do the probability and StatsModel based analytics of graphs enable diagnostics and decisions making from computations of conditional probability tables, junction trees graph, inferences, evidences or beliefs, and the potential tables? **(LO 8.3)**
- 8.11 What are the operators provided at Apache Spark GraphX architecture? What are their uses? **(LO 8.4)**

- 8.12 How does a property graph build using GraphX operators? **(LO 8.4)**
- 8.13 Describe functions of subgraph, joinVertices and aggregateMessages operators in GraphX. **(LO 8.4)**
- 8.14 What are the applications of GraphX PageRank, ConnectedComponents and Triangle Count Algorithms? **(LO 8.4)**

Practice Exercises

- 8.1 Recapitulate Practice Exercise 5.5. (i) Draw the property graph model for the product categories, names and IDs and (ii) Create GraphDB using the triples representing the graph in triple ‘subject-predicate-object’ format, and (iii) Create GraphDB in RDF; instance identifier-property name-value format;

Table of Product categories, ProductId and Product name

ProductCategory	ProductId	ProductName
Toy_Airplane	10725	Lost Temple
Toy_Airplane	31047	Propeller Plane
Toy_Airplane	31049	Twin Spin Helicopter
Toy_Train	31054	Blue Express
Toy_Train	10254	Winter Holiday Toy_Train

(LO 8.1)

- 8.2 Make an RDF format database for nodes, edges and properties given in Example 8.3. **(LO 8.1)**
- 8.3 Recapitulate Example 8.5 for property graph of database of Students, semester, subject options, grades and teachers. Create an RDF data file using the graph shown in Figure 8.4. **(LO 8.1)**
- 8.4 Create RDF and Native Data Store for *Yearly_Car_Sales* given by following triples:

Subject	Predicate	Object
Jaguar Land Rover Sale (JLRDS)	is	8000
Hexa Sale (HDS)	is	12000
Zest Sales (ZDS)	equals	160000
Nexon Sale (NDS)	equals	200000
Safari Storme Sale (SSDS)	equals	24000

(LO 8.1)

- 8.5 Draw a property graph model for the RDF created in Practice Exercise 8.4 for *Yearly_Car_Sales*. **(LO 8.1)**
- 8.6 Draw network organization for property graph shown in Figure 8.4. **(LO 8.2)**
- 8.7 Consider 12 participants in a network, represented by vertices V1, V2, ..., up to V11 and V12. Draw a figure for a network organization using graph model (i) where V2, V5, V8, V9 follows V1, (ii) where V8 to V12 follows V2, (iv) V1 to V3, V6 to V9 follows V4, (v) V3, V5 follows V8, and (vi) V10, V11, V12 follows only first neighbours. **(LO 8.2)**
- 8.8 Find the in-degrees and out-degrees of each vertex. Describe the steps to find centralities metrics

- of betweenness and closeness. Use figure drawn in Practice Exercise 8.7. **(LO 8.3)**
- 8.9 Describe steps to compute the triangles, junction trees, shortest paths, and top K-shortest paths in graph drawn in Practice Exercise 8.7 for V1, V2, ..., V11 and V12. **(LO 8.3)**
- 8.10 List the steps for creating text file data/graphx/students.txt and a set of relationships between students is given in data/graphx/stronglyConnected.txt. Use Figure 8.4. **(LO 8.4)**
- 8.11 Write the program statements in GraphX for calculating in-degrees and out-degrees of each vertex. Describe the steps in GraphX to find centralities metrics of betweenness and closeness using text files created in Practice Exercise 8.10. **(LO 8.4)**
- 8.12 List the differences between SparkGraphX and IBM System G. **(LO 8.4)**
-

1 <https://neo4j.com/product/>

2 <http://www.systemg.research.ibm.com/>

3 David Loshin, Big Data Analytics from Strategic Planning to Enterprise Integration with Tools, Techniques, NOSQL and Graph, Morgan Kaufmann, 2013

4 https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-438-algorithms-for-inference-fall-2014/lecture-notes/MIT6_438F14_Lec14.pdf

5 <http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph-builders>

6

<https://github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/graphx>

Note:

○○● Level 1 & Level 2 category

○●● Level 3 & Level 4 category

●●● Level 5 & Level 6 category

Chapter 9

Text, Web Content, Link, and Social Network Analytics

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- LO 9.1 Use the methods of text mining and machine learning (ML)— Naïve-Bayes classifier, and support vector machines for text analytics
- LO 9.2 Get knowledge and use the methods of mining the web-links, web-structure and web-contents, and analyzing the web graphs
- LO 9.3 Get knowledge and use methods of PageRanking, analysis of web-structure, and discovering hubs, authorities and communities in web-structure
- LO 9.4 Get concepts representing social networks as graphs, social network analysis methods, finding the clustering in social network graphs, evaluating the SimRank, counting triangles (cliques) and discovering the communities

RECALL FROM EARLIER CHAPTERS

Graph Data Stores consist of various interconnected data nodes (Section 3.3.5). Models of graph and graph network organization describe the entities and objects, along with their relationships, associations and properties (Sections 8.2 and 8.3). Web and social-network graphs are examples of Graph network organization.

Graph structure analytics discovers the degree of interactions, closeness, betweenness, ranks, influences, probabilities distribution, beliefs and potentials. Analysis of the community and network discovers the close-by entities and fully mesh like connected sets. Network graph analyzes centralities, and computes the PageRank of the links (Section 8.4 and 8.5).

9.1 | INTRODUCTION

Text Analytics often termed as 'text mining' refers to analyzing and extracting the meanings, patterns, correlations and structure hidden in unstructured and semi-structured textual data. Text data stores consist of strong temporal dimensions, have modularity over time and sources, such as topics and sentiments.

Methods of machine-learning are prevalent in text analytics also. For example, when a user books an air-flight ticket using a tablet or desktop, the user receives an SMS on the mobile about details of the booking and flight timings. An ML algorithm, such as *Windows Crotona* at the mobile reads and learns by itself from the SMSs received at the phone. *Crotona* uses the ML for the SMS text analysis.. Learning results in SMS alerts to the user. An alert is reminder a day before the flight. Another alert is two hours before the flight, about the need to reach the airport. Those alerts are system-generated without prior request from the user.

The reader is required to know the meaning of the following select key terms:

Vector refers to an entity with number of interrelated elements. For example, a data point consists of n-elements in an n-dimensional space, and represents a vector to that point from the origin in the space. A word is a vector of characters as the elements. Consider a vector representation of word 'McGraw-Hill', then vector $V_{MH} = [M, c, G, r, a, w, -, H, i, l, l]$. V_{MH} is vector of 11 elements (characters) that refers to word McGraw-Hill.

Feature refers to a set of properties associated with an entity, object or category. For example, feature of properties, such as description of data analysis, data cleaning, data visualization and other topics in a book on data analytics.

Category refers to a classification on the basis of set of distinct features (for example, a category of text, document, cars, toys,

students, news or fruits).

Label refers to a name assigned to a category, for example to sports-news, latest data analytics books.

Dimension refers to a number of associated values, features or states, along the distinct spaces (dimensions). For example, a sentence has a number of words, each word has a number of characters, each word may have a feature, and so the sentences are in a three-dimensional space. Two dimensions are in metric spaces, which mean values in quantifiable spaces, such as the number of words, probability of occurrences in sentences, etc. Third dimension is in feature space, measured by a feature such as noun, verb, adverb, preposition, punctuation marks and stop word.

Another example is *apples*. Metric space variables are values, such as variables n , *number of apples* of specific properties, and P the *probability of preferring apples* of specific properties. Assume, feature space variables are four properties, *colour*, *shape*, *type* and *freshness*. Metric parameters and properties of apples are said to be in six-dimensional space, two are metric space (n and P) and four are feature space.

Graph data model refers to the data modelled by a set of entities. The entities identify by vertices V . A set of relations or associations identifies by edges E . An edge e represents a relation or association between two entities. Nodes represent the entities in the graph. The model also represents a hierarchy between the parent and children nodes.

Graph data network organization refers to a structure created by organizing entities or objects in a network, such as social network, business network and student network. A network organization means where persons or entities interconnect with each other, and have areas of common interest, business or study. A graph enables ease in traversing from one entity, person or web page link to another in the network by following a path. Web graph and social network graph enable such analysis. A graph network organization models the web and social networks. Examples of social networks are SlideShare, LinkedIn, Facebook and Twitter. The analytics of social networks finds the link ranks, clusters and correlations. The analytics discovers hubs and communities.

Web content mining refers to the discovery of useful information from web documents and services. Search engines use web content mining. A search provides the links of the required information to the user.

Hyperlinks refer to links mentioned in the contents that enable the retrieval of contents at web, file, object or resources repository.

Link analytics means web structure mining of hyperlinks between web documents. The analytics of links and analyzing them for metrics such as page ranks, clusters, correlations, hubs and communities.

Count triangles Algorithm is an algorithm that finds a number of triangular relationships among the nodes. Triangular relationships mean interrelations between each other.

Graph node centrality metric means the centrality of a node in reference to other nodes using certain metrics. Metrics used for centrality of a node are degree, closeness, betweenness or other characteristics of the node, such as rank, belief, potential, expectation, evidence, reputation or status.

Degree centrality of a node refers to the number of direct connections. Having more number of direct connections is not always a better metric. Better measure is the fact that the connection directs to significant results and tell how the nodes connect to the isolated node.

Betweenness centrality is a measure that provides the extent to which a node lies on paths between other nodes. A node with high betweenness signifies high influence over what flows in the network indicating importance of link and single point of failure.

Closeness centrality is the degree to which a node is near all other nodes in a network (directly or indirectly). It reflects the ability to access information through the network.

The present Chapter focuses on text, web, contents, structure and social network graph analytics. Section 9.2 describes text mining and usage of ML techniques—Naïve-Bayes analysis and support vector machines (SVMs) for analyzing text. Section 9.3 describes web mining, methods to implement the system, and analyzing the web graphs.

Section 9.4 describes PageRank methods, web structure analytics and finding the hubs and communities. Section 9.5 describes social network analysis, representation of social networks as graphs and computational methods of finding the clustering in social network graphs, evaluating the SimRank, counting triangles (cliques) and discovering the communities.

This chapter follows a method of notations as mentioned earlier in Section 6.1 for fonts when absolute value, mean value, function value, vector element or set member, entity or variable when these denote by a character or character-set. This chapter follows the notations for the probabilities, earlier specified in

Section 8.3.2. Condition probability P specifies as $P(x_i|c_k)$ which means probability of variable $x = x_i$ at condition $c = c_k$.

9.2 ! TEXT MINING

Today, large amounts of textual data is generated in computing applications. Text stream arriving continuously over time generates text data. For example, news articles, news reports, online comments on news, online traffic reports, corporate reports, web searches, and contents at social media discussion-forums (such as LinkedIn, Twitter and Facebook), short messages on phones, chat messages, transcripts of phone conversations, blogs and e-mails.

LO 9.1

Methods of text mining and machine learning (ML)—
Naïve-Bayes classifier, and support vector machines
for text analytics

The abundance of textual data leads to problems which relate to their collection, exploration and ways of leveraging data. Textual data presents challenges for computing and storage requirements, consists of a strong temporal dimension, has modularity over time and have sources such as topics and sentiments. Examples of text processing techniques are clustering analysis, classifications, evolution analysis and event detection. Following subsections describe text mining in details:

9.2.1 Text Mining

Four definitions are:

1. “Text mining refers to the process of deriving high-quality information from text.” (Wikipedia)
2. “Text mining is the process of discovering and extracting knowledge from unstructured data.” (National Center of Text Mining –The University of Manchester¹)
3. “Text mining is the process of analyzing collections of textual contents in order to capture key concepts themes, uncover hidden relationships, and discover the trends without requiring that you know the precise words or terms that authors have used to express those concepts.” (IBM²)
4. “Text mining is a technique which helps in revealing the patterns and relationships in large volumes of textual content that are not visible to the naked eye, leading to new business opportunities and improvements in processes.” (Amazon BigData Official Blog³)

Applications of text mining in business domains are predicting stock movements from analysis of company results, decision making for product and innovations developed at the company and contextual advertising. Some other applications are (i) mail filtering (spam), (ii) drug action reports (iii) fraud detection (iv) knowledge management, and (iv) social media data analysis.

The applications provide innovative and insightful results. The results when combined with other data sources, find the answers to the following:

- (i) Two terms which occur together
- (ii) Information linkage with another information
- (iii) Different categories that can be created from extracted information
- (iv) Prediction of information or categories.

9.2.1.1 Text Mining Overview

Text mining includes extraction of high-quality information, discovering and extracting knowledge, and revealing patterns and relationships from unstructured data available in the form of text.

The term *text analytics* evolves from provisioning of strong integration with the already existing database technology, artificial intelligence, machine learning, data mining and text Data Store techniques. Information retrieval, natural language processing (NLP), classification, clustering and knowledge management are some of such useful techniques. Figure 9.1 shows process-pipeline in text-analytics.

9.2.1.2 Areas and Applications of Text Mining

Natural Language Processing (NLP) is a technique for analyzing, understanding and deriving meaning from human language. NLP involves the computer's understanding and manipulation of human language. NLP algorithms are typically based on ML algorithms. They automatically learn the rules. First, they analyze set of examples from a large collection of sentences in a book. Then, they make the statistical inferences.

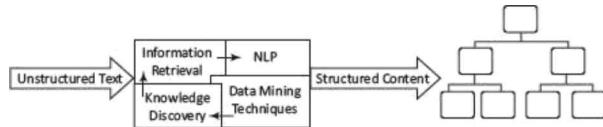


Figure 9.1 Text analytics process pipeline

NLP contributes to the field of human computer interaction by enabling several real-world applications such as automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, relationship extraction and stemming. The common uses of NLP include text mining, machine translation and automated question answering.

Information Retrieval (IR) is a process of searching and retrieving a subset of documents from the abundant collection of documents. IR can also be defined as extraction of information required by a user. IR is an area derived fundamentally from database technology. One of the most popular applications of IR is searching the information on the web. Search engines provide IR using various advance techniques. For example, the crawler program is capable of retrieving information from a wide variety of data sources. Search methods use metadata or full-text indexing.

Information Extraction (IE) is a process in which the software extracts structured information from unstructured and/or semi-structured documents. IE finds the relationship within text or desired contents from text. IE ideally derives from machine learning, more specifically from the NLP domain. Content extraction from the images, audio or video is an example of information extraction.

IE requires a dictionary of extraction patterns (For example, "Citizen of <x>, or "Located in <x>") and a semantic lexicon (dictionary of words with semantic category labels).

Document Clustering is an application which groups text documents into clusters. Automating document organization, topic extraction and fast information retrieval or filtering use the document clustering method. For example, web document clustering facilitates easy search by users.

Document Classification is an application to classify text documents into classes or categories. The application is useful for publishers, news sites, blogs or areas where lot of contents are present.

Web Mining is an application of data mining techniques. They discover patterns from the web Data Store. The patterns facilitate understanding. They improve the services of web-based applications. Data mining of web usage provides the browsing behavior of a website.

Concept Extraction is an application that deals with the extraction of concept from textual data. Concept extraction is an area of text classification in which words and phrases are classified into a semantically similar group.

9.2.1.3 Text Mining Process

Text is most commonly used for information exchange. Unlike data stored in databases, text is unstructured, ambiguous and difficult to process. Text mining is the process that analyzes a text to extract information useful for a specific purpose.

Syntactically, a text document comprises characters that form words, which can be further combined to generate phrases or sentences. Text mining steps are (i) recognizing, extracting and using the information present in words. Along with searching of words, mining involves search for semantic patterns as well.

Text mining process consists of a process-pipeline. The pipeline processes execute in several phases. Mining uses the iterative and interactive processes. The processing in pipeline does text mining efficiently and mines the new information. Figure 9.2 shows five phases of the process pipeline.

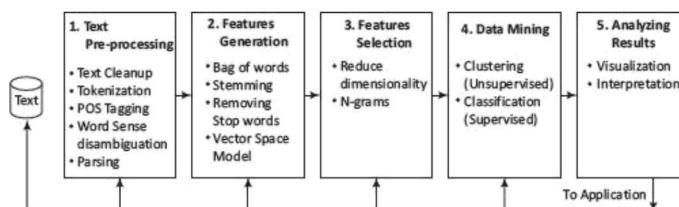


Figure 9.2 Five phases in a process pipeline

The following subsection describes these phases:

9.2.1.4 Text Mining Process Phases

The five phases for processing text are as follows:

Phase 1: Text pre-processing enables Syntactic/Semantic text-analysis and does the followings:

1. *Text cleanup* is a process of removing unnecessary or unwanted information. Text cleanup converts the raw data by filling up the missing values, identifies and removes outliers, and resolves the inconsistencies. For example, removing comments, removing or escaping "%20" from URL for the web pages or cleanup the typing error, such as teh (the), do n't (do not) [%20 specifies space in a URL].
2. *Tokenization* is a process of splitting the cleanup text into tokens (words) using white spaces and punctuation marks as delimiters.
3. *Part of Speech (POS) tagging* is a method that attempts labeling of each token (word) with an appropriate POS. Tagging helps in recognizing names of people, places, organizations and titles. English language set includes the noun, verb, adverb, adjective, prepositions and conjunctions. Part of Speech encoded in the annotation system of the Penn Treebank Project has 36 POS tags.⁴
4. *Word sense disambiguation* is a method, which identifies the sense of a word used in a sentence; that gives meaning in case the word has multiple meanings. The methods, which resolve the ambiguity of words can be context or proximity based. Some examples of such words are bear, bank, cell and bass.
5. *Parsing* is a method, which generates a parse-tree for each sentence. Parsing attempts and infers the precise grammatical relationships between different words in a given sentence.

Phase 2: Features Generation is a process which first defines features (variables, predictors). Some of the ways of feature generations are:

1. *Bag of words*—Order of words is not that important for certain applications.

Text document is represented by the words it contains (and their occurrences). Document classification methods commonly use the bag-of-words model. The pre-processing of a document first provides a document with a bag of words. Document classification methods then use the occurrence (frequency) of each word as a feature for training a classifier. Algorithms do not directly apply on the bag of words, but use the frequencies.

2. *Stemming*—identifies a word by its root.

- (i) Normalizes or unifies variations of the same concept, such as *speak* for three variations, i.e., speaking, speaks, speakers denoted by [speaking, speaks, speaker → speak]
 - (ii) Removes plurals, normalizes verb tenses and remove affixes.

Stemming reduces the word to its most basic element. For example, impurification → pure.

3. *Removing stop words* from the feature space—they are the common words, unlikely to help text mining. The search program tries to ignore stop words. For example, ignores *a, at, for, it, in* and *are*.
4. *Vector Space Model (VSM)*—is an algebraic model for representing text documents as vector of identifiers, word frequencies or terms in the document index. VSM uses the method of term frequency-inverse document frequency (TF-IDF) and evaluates how important is a word in a document.

When used in document classification, VSM also refers to the bag-of-words model. This bag of words is required to be converted into a term-vector in VSM. The term vector provides the numeric values corresponding to each term appearing in a document. The term vector is very helpful in feature generation and selection.

Term frequency and inverse document frequency (IDF) are important metrics in text analysis. TF-IDF weighting is most common—Instead of the simple TF, IDF is used to weight the importance of word in the document.

TF-IDF stands for the 'term frequency-inverse document frequency'. It is a numeric measure used to score the importance of a word in a document based on how often the word appears in that document and in a given collection of documents. It suggests that if a word appears frequently in a document, then it is an important word, and should therefore be high in score. But if a word appears in many more other documents, it is probably not a unique identifier, and therefore should be assigned a lower score. The TF-IDF is measured as:

$$\text{TF-IDF}(t) = \frac{\text{No. of times } t \text{ appears in a document}}{\text{Total No. of terms in the document}} \times \log \frac{\text{No. of documents in the collection}}{\text{No. of documents that contain } t} \quad (9.1)$$

where t denotes the term vector.

Following example suggests method of calculating TF-IDF (t):

EXAMPLE 9.1

Consider a document containing 1000 words wherein the word toys appears 16 times. How will the TF-IDF weight be calculated?

SOLUTION

The term frequency (TF) for toys is then $(16/1000) = 0.016$. Let, there are 10 million documents and the word toys appear in 1000 of them. Then, the inverse document frequency (IDF) is calculated as $\log_{10} (10,000,000/1,000) = 4$.

$$\text{TF-IDF weight} = 0.016 \times 4 = 0.064$$

Additional weight is assigned to terms appearing as keywords or in titles. Documents are usually represented as a sparse vector of terms weights and extra weights are added to the terms appearing in title or keywords.

Pre-processing of web data succeeds the conversion of bag of words into vector space model (VSM) or simply by vector creation.

Common Information Retrieval Technique – Vector space model (VSM) is an algebraic model for representing textual information as vectors of identifiers, such as, index terms. Information retrieval methods use VSM.

Each document or HTML page represents by a sparse vector of term weights. The sparse matrices represent the term frequencies (TFs).

(Sparse vector and sparse-matrix have many elements as zero or null. An associated metadata enables data storing of them in a form which does not include zeros in case of large datasets. The metadata then includes indices map with the positions in the list of elements of the vector or matrix.)

The following example gives the conversion method for evaluating TFs and matrix in pre-processing phase.

EXAMPLE 9.2

Assume that the documents below define the document space with five documents d1, d2, d3, d4 and d5:

Train Document Set:

d1: Children like the toys.

d2: The toys are precious.

Test Document Set:

d3: There are many toys in the shop.

d4: Some toys are precious and some toys are costly as well.

d5: The toys shop is one of the famous shops.

How will be the documents term vector and matrix be calculated for features generation/selection?

SOLUTION

First, create an index vocabulary of the words of the train document set using the documents d1 and d2 from the document set. The index vocabulary E(t) where t is the term will be:

$$E(t) = \begin{cases} 1. & \text{when } t = \text{"children"} \\ 2. & \text{when } t = \text{"toys"} \\ 3. & \text{when } t = \text{"precious"} \\ 4. & \text{when } t = \text{"shop"} \\ 5. & \text{when } t = \text{"costly"} \\ 6. & \text{when } t = \text{"famous"} \end{cases}$$

Note that the stop words are already not considered during the pre-processing step. The term frequency (TF) is a measure of how many times the terms present in vocabulary E(t) are present in the documents d3, d4 and d5.

$$TF(t, d) = \sum_{x \in d} \text{count}(x, t) \quad (9.2)$$

where the count (x, t), is a simple function defined as:

$$\text{count}(x, t) = \begin{cases} 1, & \text{if } x = t \\ 0, & \text{otherwise} \end{cases} \quad (9.3)$$

For example, TF ("toys", d4) = 2.

Create the document vector as:

$$\overline{v_{dn}} = (TF(t1, dn), TF(t2, dn), \dots, TF(tm, dn)) \quad (9.4)$$

Thus, the documents d3, d4 and d5 are represented as vector as:

$$\begin{aligned} \overline{v_{d3}} &= (TF(t1, d3), TF(t2, d3), \dots, TF(tm, d3)) \\ \overline{v_{d4}} &= (TF(t1, d4), TF(t2, d4), \dots, TF(tm, d4)) \\ \overline{v_{d5}} &= (TF(t1, d5), TF(t2, d5), \dots, TF(tm, d5)) \end{aligned} \quad (9.5)$$

This gives:

$$\begin{aligned} \overline{v_{d3}} &= (1, 1, 0, 1, 0, 0) \\ \overline{v_{d4}} &= (0, 2, 1, 0, 1, 0) \\ \overline{v_{d5}} &= (0, 1, 0, 2, 0, 1) \end{aligned} \quad (9.6)$$

The resulting vector $\overline{v_{d3}}$ shows 1 occurrence of the term "children", 1 occurrence of the term "toys" and so on. In the $\overline{v_{d4}}$, there is 0 occurrence of the term "children", 2 occurrences of the term "toys" and so on.

A collection of web documents requires representation as vectors. Another representation is a matrix with $|D| \times F$ shape, where $|D|$ is the cardinality of the document space (total number of documents) and the F is the number of features. F represents the vocabulary size in the example. Matrix representation of the vectors described above is by 3×6 matrix as follows:

$$M_{|D| \times F} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \end{bmatrix} \quad (9.7)$$

Example 9.2 shows that the matrices representing term frequencies tend to be very sparse (with majority of terms zeroed). A common representation of such matrix is thus the sparse matrices.

Phase 3: Features Selection is the process that selects a subset of features by rejecting irrelevant and/or redundant features (variables, predictors or dimension) according to defined criteria. Feature selection process does the following:

1. *Dimensionality reduction*—Feature selection is one of the methods of division and therefore, dimension reduction. The basic objective is to eliminate irrelevant and redundant data. Redundant features are those, which provide no extra information. Irrelevant features provide no useful or relevant information in any context.

Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) are dimension reduction methods. Discrimination ability of a feature measures relevancy of features. Correlation helps in finding the redundancy of the feature. Two features are redundant to each other if their values correlate with each other.

2. *N-gram evaluation*—finding the number of consecutive words of interest and extract them. For example, 2-gram is a two words sequence, ["tasty food", "Good one"]. 3-gram is a three words sequence, ["Crime Investigation Department"].
3. *Noise detection and evaluation of outliers* methods do the identification of unusual or suspicious items, events or observations from the data set. This step helps in cleaning the data.

The feature selection algorithm reduces dimensionality that not only improves the performance of learning algorithm but also reduces the storage requirement for a dataset. The process enhances data understanding and its visualization.

Phase 4: Data mining techniques enable insights about the structured database that resulted from the previous phases. Examples of techniques are:

1. Unsupervised learning (for example, clustering)
 - (i) The class labels (categories) of training data are unknown
 - (ii) Establish the existence of groups or clusters in the data

Good clustering methods use high intra-cluster similarity and low inter-cluster similarity. Examples of uses – blogs, patterns

and trends.

2. *Supervised learning (for example, classification)*

- (i) The training data is labeled indicating the class
- (ii) New data is classified based on the training set

Classification is correct when the known label of test sample is identical with the resulting class computed from the classification model.

Examples of uses are *news filtering application*, where it is required to automatically assign incoming documents to pre-defined categories; *email spam filtering*, where it is identified whether incoming email messages are spam or not.

Example of text classification methods are *Naïve Bayes Classifier* and *SVMs*.

3. *Identifying evolutionary patterns* in temporal text streams—the method is useful in a wide range of applications, such as summarizing of events in news articles and extracting the research trends in the scientific literature.

Phase 5: Analysing results

- (i) Evaluate the outcome of the complete process.
- (ii) Interpretation of Result– If acceptable then results obtained can be used as an input for next set of sequences. Else, the result can be discarded, and try to understand what and why the process failed.
- (iii) Visualization – Prepare visuals from data, and build a prototype.
- (iv) Use the results for further improvement in activities at the enterprise, industry or institution.

Open source tools, such as *nltk* are available for text analytics. Online contents accompanying book describe how text analytics tasks can be performed using Python library *nltk* in the solution of Practice Exercise 9.2.

9.2.1.5 Text Mining Challenges

The challenges in the area of text mining can be classified on the basis of documents area-characteristics. Some of the classifications are as follows:

- 1. NLP issues:
 - (i) POS Tagging
 - (ii) Ambiguity
 - (iii) Tokenization
 - (iv) Parsing
 - (v) Stemming
 - (vi) Synonymy and polysemy
- 2. Mining techniques:
 - (i) Identification of the suitable algorithm(s)
 - (ii) Massive amount of data and annotated corpora
 - (iii) Concepts and semantic relations extraction
 - (iv) When no training data is available
- 3. Variety of data:
 - (i) Different data sources require different approaches and different areas of expertise
 - (ii) Unstructured and language independency
- 4. Information visualization
- 5. Efficiency when processing real-time text stream
- 6. Scalability

9.2.1.6 Supervised Text Classification

The categorization of text documents requires information retrieval, ML and NLP techniques. Some important approaches to

automatic text categorization are based on ML techniques.

The *supervised text classification* requires labeled documents and additional knowledge from experts. The algorithms exploit the training data (where zero or more categories) to learn a classifier, which classifies new text documents and labels each document. A document is considered as a positive example for all categories with which it is labeled, and as a negative example to all others. The task of a training algorithm for a text classifier is to find a weight vector which best classifies new text documents.

The different approaches for supervised text classification are:

- (i) K-Nearest Neighbour Method
- (ii) Support Vector Machine
- (iii) Naïve Bayes Method
- (iv) Decision Tree
- (v) Decision Rule

K-Nearest Neighbours (KNN) method makes use of training text document. The training documents are the previously categorized set of documents. They train the system to understand each category. The classifier uses the training ‘model’ to classify new incoming documents. KNN assumes that close-by objects are more probable in the same category. KNN finds k objects in the large number of text documents, which have most similar query responses. Thus, in KNN, predictions are based on a method that is used to predict new (not observed earlier) text data. The predictions are by (i) majority vote method (for classification tasks) and (ii) averaging (for regression) method over a set of K-nearest examples.

The decision trees or decision rules are built to predict the category for an input document. A decision tree or rule represents a set of nested logical if-then conditions on the observed values of the text features that enable the prediction of the target variable. The decision tree and decision rules are also used to classify (categorize) the document. Classification is done by recursively splitting the text features into a set of non-overlapping regions (Refer Section 6.8). (Section 6.7.4)

The following subsections describe Naïve Bayes Method and Support Vector Machines in detail.

9.2.2 Naïve Bayes Analysis

Naïve Bayes classifier is a simple, probabilistic and statistical classifier. It is one of the most basic text classification techniques, also known as multivariate Bernoulli method. Naïve Bayes classifies using Bayes theorem along with the Naïve independence assumptions (conditional independence). The classifier computes condition probabilities for the conditional independence (Refer Section 8.3.2).

Probability that a bag-of-words \mathbf{x} belong to k^{th} class equals the product of individual probabilities of those words. $P(\mathbf{x}|c_k) = \prod P(x_i|c_k)$, where x_i is a discrete random variable (word), $i = 1, 2, \dots, n$, where n is number of words in the bag. \prod is sign for the product of n terms. $[P(x_i|c_k)$ means probability of condition that state the value $= x_i$ and of $c = c_k$ (Example 8.6)].

The $P(\mathbf{x}|c_k)$ is normalized as all distributed probabilities equals 1. $P(\mathbf{x}|c_k)$ is normalized by dividing the product on right hand side by $\sum_i P(x_i|c_k) P(c_k)$.

The following example gives the method of deciding the most likely class.

EXAMPLE 9.3

How is “maximum a posteriori (MAP)” used to obtain the most likely class and take a decision?

SOLUTION

Text classification problem uses the words (or tokens) of the document in order to classify it on the appropriate class. Bayes’ rule is applied to documents and classes. For a document (d) and class (c), we get:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (9.8)$$

The “maximum a posteriori (MAP)” (to obtain most likely class) decision rule is applied to documents and classes:

$$c_{\text{MAP}} = \operatorname{argmax}_{c \in C} P(c|d) \quad (9.9)$$

(MAP is “maximum posteriori” = most likely class)

$$c_{MAP} = \operatorname{argmax}_{c \in C} \left(\frac{P(d|c)P(c)}{P(d)} \right) \quad (\text{Bayes Rule}) \quad (9.10)$$

$$c_{MAP} = \operatorname{argmax}_{c \in C} (P(d|c)P(c)) \quad (\text{Dropping the denominator}) \quad (9.11)$$

$$c_{MAP} = \operatorname{argmax}_{c \in C} (P(t_1, t_2, \dots, t_n | c)P(c)). \quad (9.12)$$

where t_1, t_2, \dots, t_n are tokens of document.

Multinomial Naïve Bayes independence assumptions

$$P(t_1, t_2, \dots, t_n | c) \quad (9.13)$$

Bag-of-words assumption: Assume the position of the word does not matter.

Conditional independence: Assume the feature probabilities $P(t_i | c_j)$, are independent given the class(c):

$$P(t_1, t_2, \dots, t_n | c) = P(t_1 | c) * P(t_2 | c) * P(t_3 | c) * \dots * P(t_n | c) \quad (9.14)$$

and thus, conditional independences are given by

$$c_{MAP} = \operatorname{argmax}_{c \in C} (P(t_1, t_2, \dots, t_n | c)P(c)) \quad (9.15a)$$

$$c_{NB} = \operatorname{argmax}_{c \in C} \left(P(c_j) \prod_{t \in T} P(t | c) \right) \quad (9.15b)$$

Applying multinomial Naïve Bayes classifier to text classification where positions " all word positions in the text document,

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \left(P(c_j) \prod_{i \in \text{Positions}} P(t_i | c_j) \right) \quad (9.16)$$

The equation estimates the product of the probability of each word of the document given a particular class (likelihood), multiplied by the probability of the particular class (prior) to find in which class one should classify a new document.

Select the one with the highest probability among all the classes of set C . Calculation of product of the probabilities leads to float point underflow when handling numbers with specific decimal point accuracy by computing devices. Such small numbers will be rounded to zero, implying the analysis is of no use at all. In order to avoid this, instead of maximizing the product of the probabilities, the maximization of the sum of their logarithms is done:

$$c_{NB} = \operatorname{argmax}_{c_j \in C} \left(\log P(c_j) \prod_{i \in \text{Positions}} \log P(t_i | c_j) \right) \quad (9.17)$$

Here, choose the one with the highest log score rather than choosing the class with the highest probability. Given that the logarithm function is monotonic, the decision of MAP remains the same.

When compared with other techniques, such as Random Forest, Max Entropy and SVM, the Naïve Bayes classifier performs efficiently in terms of less CPU and memory consumption. Naïve Bayes classifier requires a small amount of training data to estimate the parameters. The classifier is not sensitive to irrelevant features as well. Furthermore, the training time is significantly smaller with Naïve Bayes as opposed to other techniques.

The classifier is popularly used in a variety of applications, such as email spam detection, personal email sorting, document categorization, language detection, authorship identification, age/gender identification and sentiment detection.

9.2.3 Support Vector Machines

Support vector machines (SVM) is a set of related *supervised learning methods* (the presence of training data) that analyze data, recognize patterns, classify text, recognize hand-written characters, classify images, as well as bioinformatics and bio sequence analysis.

A vector has in general n components, x_2, x_3, \dots, x_n . A datapoint represents by (X_1, X_2, \dots, X_n) in n -dimensional space. Assume for the sake of simplicity, that a vector has two components, X_1 and X_2 (Two sets of words in text analysis).

Section 6.7.6 described the use of the concept of hyperplanes for classification. A *hyperplane* is a subspace of one dimension less than its ambient space in geometry (Figure 6.18). If a space is 3-dimensional then its hyperplanes are 2-dimensional planes, while if the space is 2-dimensional, its hyperplanes are 1-dimensional, which means lines.

The hyperplane which separates the two classes most appropriately has maximum distance from closest data points of the distinct classes. This distance is termed as margin. Figure 9.3 shows the concept of support vectors, separating hyperplane and margins when using B as a classifier. The margin for hyperplane B in Figure 9.3 is more as compared to two hyperplanes, A and C shown by dotted lines. The margin of the data points from B is maximum. Therefore, the hyperplane B is the **maximum margin classifier**.

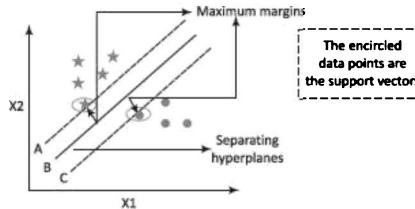


Figure 9.3 Support vectors, separating hyperplane (B) and margins

A and C are closest (least margins) to the data points. These are called the **support vectors**. They support the classifications of the star and dotted data points. [Remember that with n-dimensional datapoints space, a hyperplane has the vectors along (n - 1) axes.]

The support vectors are such that a set of data points lies closest to the decision (classification) surface (or hyperplane). Those points are most difficult to classify. They have direct bearing on the optimum location of the classification surface. Support vectors along maximum margin classification surface are thus gives the best results.

Thus, a SVM classifier is a **discriminative classifier** formally defined by a separating hyperplane. The concept applies extensively in number of application areas of ML. Applications of SVMs are as follows:

1. Classification based on the outputs taking discrete values in a set of possible categories, SVM can be used to separate or predict if something belongs to a particular class or category. SVM helps in finding a decision boundary between two categories.
2. Regression analysis, if learning problem has continuous real-valued output (continuous values of x, in place discrete n values, $(x_1, x_2, x_3, \dots, x_n)$)
3. Pattern recognition
4. Outliers detection.

The following example illustrates the discriminative classifier method, formally defined by a separating a hyperplane for taking the decision for effective elements (entities, set of words, itemsets) in a training set.

EXAMPLE 9.4

How is the discriminative classifier used?

SOLUTION

Consider a mapping function (f) used for linear separation in the feature space (H). An optimal separating hyperplane depends on the data through dot products in H ($f(x_i) \cdot f(x_j)$).

A kernel function k is required that acts as a measure of similarity. Let us use k where

$$k(x_i, x_j) = f(x_i) \cdot f(x_j) \quad (9.18)$$

The objective is to select the hyperplane which separates the two classes most appropriately. This helps in identifying the right or appropriate hyperplane. Figure 9.3 showed three hyperplanes. A, B and C. A and C are least margin planes and B is maximum margin plane.

A hyperplane,maximum margin classifier is the right hyperplane. It has maximum distances from the nearest data points (of either classes). An important reason for selecting the maximum margin classification surface is robustness. A hyperplane having low margin has considerably high chance of misclassifying.

Binary Classification

For a given training data $[x_i, y(x_i)]$ for $i = 1 \dots N$, with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such that:

$$f(x_i) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases} \quad (9.19)$$

The above equation implies that $y_i f(x_i) > 0$ for correct classification.

Figure 9.4 shows a two-class classification. The method is using one hyperplane B for separating two-class classification of data points.

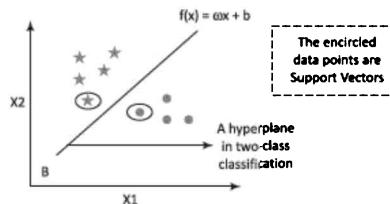


Figure 9.4 Concept of training set data using support vectors

Let us take the simplest case of two-class classification. Suppose there are two features X_1 and X_2 and it is required to classify objects as shown in the Figure 9.4. Stars and dots represent the objects (itemsets, sets of words, entities) of two classes. The goal is to design a hyperplane (B) that classifies all training vectors in two classes for linearly separable binary set.

The following example gives the method to design a hyperplane (B) that classifies all training vectors:

EXAMPLE 9.5

How will you select an appropriate hyperplane that classifies all training vectors?

SOLUTION

Plane B is $f(x) = wx + b$ where w is weight vector. Figure 9.5 shows the method of selecting the right hyperplane.

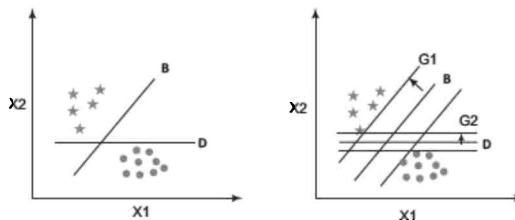


Figure 9.5 Method of selecting the right hyperplane

The best choice is the hyperplane that leaves the maximum margin from both the classes. Thus, B is the right choice, since $G_1 > G_2$.

Thus, for line segment B,

$$\begin{aligned} f(x) &\geq 1 \quad \forall x \rightarrow \text{class 1} \\ f(x) &\leq -1 \quad \forall x \rightarrow \text{class 2} \end{aligned} \quad (9.20)$$

Self-Assessment Exercise linked to LO 9.1

1. Define text analytics.
2. List the steps in text pre-processing phase. Why are tokenization and POS tagging needed? Give an example of each step.
3. How is bag-of-word used in text analysis? Give 5 examples of stemming the affix wordforms to its root word.
4. How do the TF-IDF weighting and sparse matrices represent the term frequencies (TFs) for use in text analysis?
5. How does maximum a posteriori (MAP) in Naïve Bayes classifier enable the decision about classification?
6. How does support vectors in SVMs classify the data points in n-dimensional space.

9.3 WEB MINING, WEB CONTENT AND WEB USAGE ANALYTICS

Web is a collection of interrelated files at web servers. Web data refers to (i) web content—text, image and records, (ii) web structure—hyperlinks and tags, and (iii) web usage—http logs and application server logs.

LO 9.2

Mining the web-links, web-structure and web-contents, and analyzing the web graphs

Features of web data are:

1. Volume of information and its ready availability
2. Heterogeneity
3. Variety and diversity (Information on almost every topic is available using different forms, such as text, structured tables and lists, images, audio and video.)
4. Mostly semi-structured due to the nested structure of HTML code
5. Hyperlinks among pages within a website, and across different websites
6. Redundant or similar information may be present in several pages
7. Mostly, the web page has multiple sections (divisions), such as main contents of the page, advertisements, navigation panels, common menu for all the pages of a website and copyright notices
8. A web form or HTML form on a web page enables a user to enter data that is sent to a server for processing
9. Website contents are dynamic in nature where information on the web pages constantly changes, and fast information growth takes place such as conversations between users, social media, etc.

The following subsections describe web data mining and analysis methods:

9.3.1 Web Mining

Data Mining is a process of discovering patterns in large datasets to gain knowledge. The process can be shown as [Raw Data → Patterns → Knowledge]. Web data mining is the mining of web data. Web mining methods are in multidisciplinary domains: (i) data mining, ML, natural language, (ii) processing, statistics, databases, information retrieval, and (iii) multimedia and visualization.

Web consists of rich features and patterns. A challenging task is retrieving interesting content and discovering knowledge from web data. Web offers several opportunities and challenges to data mining.

Definition of Web Mining

Web mining refers to the use of techniques and algorithms that extract knowledge from the web data available in the form of web documents and services. Web mining applications are as follows:

- (i) Extracting the fragment from a web document that represents the full web document
- (ii) Identifying interesting graph patterns or pre-processing the whole web graph to come up with metrics, such as PageRank
- (iii) User identification, session creation, malicious activity detection and filtering, and extracting usage path patterns

Web Mining Taxonomy

Web mining can broadly be classified into three categories, based on the types of web data to be mined. Three ways are web content mining, web structure mining and web usage mining. Figure 9.6 shows the taxonomy of web mining.

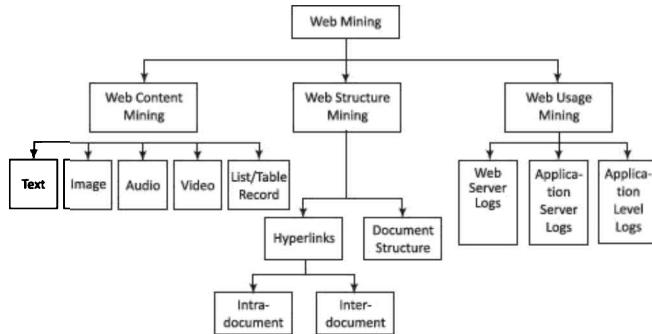


Figure 9.6 Web mining taxonomy

Web content mining is the process of extracting useful information from the contents of web documents. The content may consist of text, images, audio, video or structured records, such as lists and tables.

Web structure mining is the process of discovering structure information from the web. Based on the kind of structure-information present in the web resources, web structure mining can be divided into:

1. Hyperlinks: the structure that connects a location at a web page to a different location, either within the same web page (intra-document hyperlink) or on a different web page (inter-document hyperlink)
2. Document Structure: The structure of a typical web graph consists of web pages as nodes, and hyperlinks as edges connecting the related pages.

Web usage mining is the application of data mining techniques which discover interesting usage patterns from web usage data. The data contains the identity or origin of web users along with their browsing behavior at a web site. Web usage mining can be classified as:

- (i) Web Server logs: Collected by the web server and typically include IP address, page reference and access time.
- (ii) Application Server Logs: Application servers typically maintain their own logging and these logs can be helpful in troubleshooting problems with services.
- (iii) Application Level Logs: Recording events usually by application software in a certain scope in order to provide an audit trail that can be used to understand the activity of the system and to diagnose problems.

9.3.2 Web Content Mining

Web Content Mining is the process of information or resource discovery from the content of web documents across the World Wide Web. Web content mining can be (i) direct mining of the contents of documents or (ii) mining through search engines. They search fast compared to direct method.

Web content mining relates to both, data mining as well as text mining. Following are the reasons:

- (i) The content from web is similar to the contents obtained from database, file system or through any other mean. Thus, available data mining techniques can be applied to the web.
- (ii) Content mining relates to text mining because much of the web content comprises texts.
- (iii) Web data are mainly semi-structured and/or unstructured, while data mining is structured and the text is unstructured.

Applications

Following are the applications of content mining from web documents:

1. Classifying the web documents into categories
2. Identifying topics of web documents
3. Finding similar web pages across the different web servers
4. Applications related to relevance:
 - (a) Recommendations – List of top “n” relevant documents in a collection or portion of a collection

- (b) Filters – Show/Hide documents based on some criterion
- (c) Queries – Enhance standard query relevance with user, role, and/or task-based relevance.

9.3.2.1 Common Web Content Mining Techniques

Pre-processing of contents The pre-processing steps are quite similar to the pre-processing for text mining. The content preparation involves:

1. Extraction of text from HTML
2. Data cleaning by filling up the missing values and smoothing the noisy data
3. *Tokenizing*: Generates the tokens of words from the cleaned up text
4. *Stemming*: Reduce the words to their roots. The different grammatical forms or declensions of verbs identify and index (count) as the same word. For example, stemming will ensure that both “closed” and “closing” are derived from the same word “close”. Stemming algorithm, *Porter*, can be used here. The java code for *Porter* stemming algorithm can be obtained from <https://tartarus.org/martin/PorterStemmer/java.txt>,
5. *Removing the stop words*: The common words unlikely to help in the mining process such as articles (a, an, the), or prepositions (such as, to, in, for) are removed.
6. *Calculate collection wide-word frequencies*: The distinct-word stem obtained after stemming process and removing the stop words results into a list of significant words (or terms). Calculating the occurrence of a significant term (t) in a collection is called collection frequency (CF_t). CF counts the multiple occurrences.)

Now, find the number of documents in the collection that contains the specific term (t). This numeric measure is the document frequency (DF_t).

7. *Calculate per Document Term Frequencies (TF)*. TF is a numeric measure that is used to score the importance of a word in a document based on how often it appeared in that document (Refer Example 9.1).
8. *Bag of words*: Web document is represented by the words it contains (and their occurrences).

The following example explains the concept of CF and DF using the data of toy sales collection.

EXAMPLE 9.6

Using the table below on collection and document frequencies, which is prepared from the toys sales collection, analyze the numeric values of CFs and DFs.

Collection frequency (CF) and document frequency (DF)

Word	CF	DF
discount	15565	550
sale	15567	1230

SOLUTION

The table suggests that the collection frequency (CF) and document frequency (DF) can behave differently. The CF values for both *discount* and *sale* are nearly equal, but their DF values differ significantly. The reason is that the word *sale* is present in a large number of documents and the word *discount* in a less number of documents. Thus, when a query related to *discount* is generated, it must be searched in the concerned documents only.

Mining Tasks for Web Content Analytics

Following are the tasks for web content analytics:

1. *Classification* – A supervised technique which:
 - (i) Identifies the class or category a new web documents belongs to from the set of predefined classes or categories
 - (ii) Categories in the form of a term vector that are produced during a “training” phase
 - (iii) Employs algorithms using term vector to categorize the new data according to the observations at the training set.
2. *Clustering* – An unsupervised technique:

- (i) Groups the web documents (clustered) with similar features using some similarity measure
 - (ii) Uses no pre-defined perception of what the groups should be
 - (iii) Measures most common similarity using the dot product between two web document vectors.
3. *Identifying the association* between web documents – Association rules help to identify correlation between web pages that occur mostly together.

The other significant mining tasks are:

1. *Topic identification, tracking and drift analysis* – A way of organizing the large amount of information retrieved from the web is categorizing the web pages into distinct topics. The categorization can be based on a similarity metric, which includes textual information and co-citation relations. Clustering or classification techniques can automatically and effectively identify relevant topics and add them in a topic-wise collection library.

Adding a new document to a collection library includes:

- (i) Assigning each document to an existing topic (category)
 - (ii) Re-checking of collection for the emergence of new topics
 - (iii) Tracking the number of views to a collection
 - (iv) Identifying the drift in a topic(s)
2. *Concept hierarchy creation* – Concept hierarchy is an important tool for capturing the general relationship among web documents. Creation of concept hierarchies is important to understand a category and sub-categories to which a document belongs. The clustering algorithms leverage more than two clusters, which merge into a cluster. That is merging the sub-clusters into a cluster.

Important factors for creation of concept hierarchy include:

- (i) Identifying the organization of categories, such as flat, tree or network
 - (ii) Planning the maximum number of categories per document
 - (iii) Building category dimensions, such as domain, location, time, application and privileges.
3. *Relevance of content* – Relevance or the applicability of web content can be measured with respect to any of the following basis:
- (i) Document relevance describes the usefulness of a given document in a specified situation.
 - (ii) Query-based relevance is the most useful method to assess the relevance of web pages. Query-based relevance is used in information retrieval tools such as search engines. The method calculates the similarity between query (search) keywords and document. Similarity, results can be refined through additional information such as popularity metric as seen in Google or the term positions in AltaVista.
 - (iii) User-based relevance is useful in personal aspects. User profiles are maintained, and similarity between the user profile and document is calculated. The relevance is often used in push notification services.
 - (iv) Role/task-based relevance is quite similar to user-based relevance. Instead of a user, here the profile is based on a particular role or task. Multiple users can provide input to profile.

9.3.3 Web Usage Mining

Web usage mining discovers and analyses the patterns in click streams. Web usage mining also includes associated data generated and collected as a consequence of user interactions with web resources.

Figure 9.7 shows three phases for web usage mining.

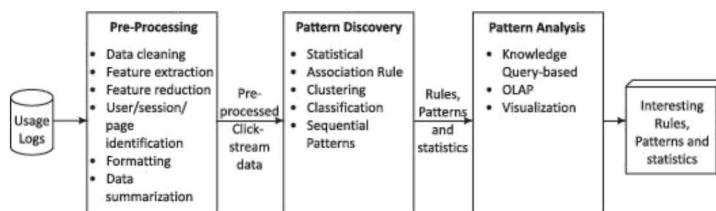


Figure 9.7 Process of web usage mining

The phases are:

1. Pre-processing – Converts the usage information collected from the various data sources into the data abstractions necessary for pattern discovery.
2. Pattern discovery – Exploits methods and algorithms developed from fields, such as statistics, data mining, ML and pattern recognition.
3. Pattern analysis – Filter outs uninteresting rules or patterns from the set found during the pattern discovery phase.

Usage data are collected at server, client and proxy levels. The usage data collected at the different sources represent the navigation patterns of the overall web traffic. This includes single-user, multi-user, single-site access and multi-site access patterns.

9.3.3.1 Pre-processing

The common data mining techniques apply on the results of pre-processing using vector space model (Refer Example 9.2).

Pre-processing is the data preparation task, which is required to identify:

- (i) User through cookies, logins or URL information
- (ii) Session of a single user using all the web pages of an application
- (iii) Content from server logs to obtain state variables for each active session
- (iv) Page references.

The subsequent phases of web usage mining are closely related to the smooth execution of data preparation task in pre-processing phase. The process deals with (i) extracting of the data, (ii) finding the accuracy of data, (iii) putting the data together from different sources, (iv) transforming the data into the required format and (iv) structure the data as per the input requirements of pattern discovery algorithm.

Pre-processing involves several steps, such as data cleaning, feature extraction, feature reduction, user identification, session identification, page identification, formatting and finally data summarization.

9.3.3.2 Pattern Discovery

The pre-processed data enable the application of knowledge extraction algorithms based on statistics, ML and data mining algorithms. Mining algorithms, such as path analysis, association rules, sequential patterns, clustering and classification enable effective processing of web usages. The choice of mining techniques depends on the requirement of the analyst. Pre-processed data of the web access logs transform into knowledge to uncover the potential patterns and are further provided to pattern analysis phase.

Some of the techniques used for pattern discovery of web usage mining are:

Statistical techniques They are the most common methods which extract the knowledge about users. They perform different kinds of descriptive statistical analysis (frequency, mean, median) on variables such as page views, viewing time and length of path for navigational.

Statistical techniques enable discovering:

- (i) The most frequently accessed pages
- (ii) Average view time of a page or average length of a path through a site
- (iii) Providing support for marketing decisions

Association rule The rules enable relating the pages, which are most often referenced together in a single server session. These pages may not be directly connected to one another using the hyperlinks.

Other uses of association rule mining are:

- (i) Reveal a correlation between users who visited a page containing similar information. For example, a user visited a web page related to admission in an undergraduate course to those who search an eBook related to any subject.
- (ii) Provide recommendations to purchase other products. For example, recommend to user who visited a web page related to a book on data analytics, the books on ML and Big Data analytics also.

- (iii) Provide help to web designers to restructure their websites.
- (iv) Retrieve the documents in prior in order to reduce the access time when loading a page from a remote site.

Clustering is the technique that groups together a set of items having similar features. Clustering can be used to:

- (i) Establish groups of users showing similar browsing behaviors
- (ii) Acquire customer sub-groups in e-commerce applications
- (iii) Provide personalized web content to users
- (iv) Discover groups of pages having related content. This information is valuable for search engines and web assistance providers.

Thus, user clusters and web-page clusters are two cases in the context of web usage mining. Web page clustering is obtained by grouping pages having similar content. User clustering is obtained by grouping users by their similarity in browsing behavior.

Model-based or distance-based clustering can be applied on web usage logs. The model type is often specified theoretically with model-based clustering. The model selection techniques and parameters estimate using maximum likelihood algorithms, such as Expectation Maximization (EM) determines the structure of model. Distance-based clustering measures the distance between pairs of web pages or users, and then groups the similar ones together into clusters. The most popular distance-based clustering techniques include partitional clustering and hierarchical clustering (Section 6.6.3).

Classification The method classifies data items into predefined classes. Classification is useful for:

- (i) Developing a profile of users belonging to a particular class or category
- (ii) Discovery of interesting rules from server logs. For example, 3750 users watched a certain movie, out of which 2000 are between age 18 to 23 and 1500 out of these lives in metro cities.

Classification can be done by using supervised inductive learning algorithms, such as decision tree classifiers, Naïve Bayesian classifiers, k-nearest neighbour classifiers, support vector machines.

Sequential pattern discovery User navigation patterns in web usage data gather web page trails that are often visited by users in the order in which pages are visited. Markov Model can be used to model navigational activities in the website. Every page view in this model can be represented as a state. Transition probability between two states can represent the probability that a user will navigate from one state to the other. This representation allows for the computation of a number of significant user or site metrics that can lead to useful rules, pattern, or statistics.

9.3.3.3 Pattern Analysis

The objective of pattern analysis is to filter out uninteresting rules or patterns from the rules, patterns or statistics obtained in the pattern discovery phase.

The most common form of pattern analysis consists of:

- (i) A knowledge query mechanism such as SQL
- (ii) Another method is to load usage data into a data cube in order to perform Online Analytical Processing (OLAP) operations
- (iii) Visualization techniques, such as graphing patterns or assigning the colors to different values, can often highlight overall patterns or trends in the data
- (iv) Content and structure information can filter out patterns containing pages of a certain usage type, content type or pages that match a certain hyperlink structure.

Data cube enables visualizing data from different angles. For example, toys data visualization using category, colour and children preferences. Another example, news from category, such as sports, success stories, films or targeted readers (children, college students, etc).

Self-Assessment Exercise linked to LO 9.2

1. Define web mining. Discuss the broad classifications of web mining and their applications.
2. List the tasks in pre-processing of web contents.
3. How are web-content mining tasks performed using machine learning algorithms?
4. How are topic identification, tracking and drift analysis done?

5. List and explain three phases of web-usage mining.
6. Highlight the techniques used for pattern discovery in web-usage mining giving an example of each.

9.4 PAGE RANK, STRUCTURE OF WEB AND ANALYZING A WEB GRAPH

Sections 9.2 and 9.3 described text data and web contents analysis. Hyperlinks links exist between the web contents. Link analysis finds the answers to the following:

1. Can a linked (web) page rank them higher or lower?
2. Can the links be modeled as edges of graphs, structure of web as graph network, and applied the tools same as for graph analytics?
3. Can web graph mining method analyze and find a link sending spams?
4. Does a set of links correspond to a hub? Do the links correspond to an authority?
5. Does a linked page has higher authority compared to others?

LO 9.3

PageRanking, analysis of web-structure, and discovering hubs, authorities and communities in web-structure

Links analysis applies to domains of social networks and e-mail. The following sub-sections describe the applications of link analysis:

9.4.1 Page Rank Definition

The in-degree (visibility) of a link is the measure of number of in-links from other links. The out-degree (luminosity) of a link is number of other links to which that link points.

PageRank definition according to earlier approaches

Assume a web structure of hyperlinks. Each hyperlink in-links to a number of hyperlinks and out-links to a number of pages. A page commanding higher authority (rank) has greater number of in-degrees than out-degrees. Therefore, one measure of a page authority can be in-degrees with respect to out-degrees.

PageRank refers to the authority of the page measured in terms of number of times a link is sought after.

PageRank definition according to the new approach

Earlier approach of page ranking based on in-links and out-links does not capture the relative authority (importance) of the parents. Page and co-authors (1998) defined a page ranking method,⁵ which considers the entire web in place of local neighbourhood of the pages and considers the relative authority of the parent links (over children).

9.4.2 Web Structure

Web structure models as directed-graphs network-organization. Vertex of the directed graph models an anchor. Let n = number of hyperlinks at the page U . Assume \mathbf{u} is a vector with elements u_1, u_2, \dots, u_n . Each page $Pg(\mathbf{u})$ has anchors, called hyperlinks. Page $Pg(\mathbf{v})$ consists of text document with m number of hyperlinks. \mathbf{v} is a vector with elements v_1, v_2, \dots, v_m . The m is number of hyperlinks at $Pg(\mathbf{v})$. A vertex u directs to another Page V . A page $Pg(\mathbf{v})$ may have number of hyperlinks directed by out-edges to other page $Pg(\mathbf{w})$. Consider the following hypotheses:

1. Text at the hyperlink represents the property of a vertex u that describes the destination V of the out-going edge.
2. A hyperlink in-between the pages represents the conferring of the authority.

Pages U and U' hyperlinks u and u' out-linking to Page V . Let Page U has three hyperlinks parenting three Pages, V one, W two, X two, U' one, and Y two, respectively. Figure 9.8 shows a web structure consisting of pages and hyperlinks.

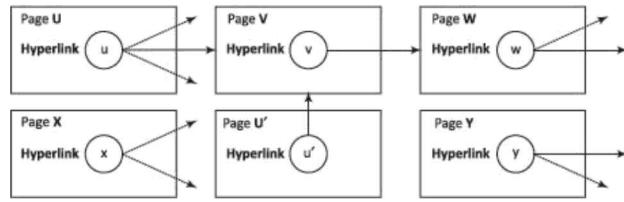


Figure 9.8 Web structure with hyperlinks from a parent to one or more pages

9.4.2.1 Dead Ends

Dead-end web pages refer to pages with no out-links. When a web page links to such pages, its page rank gets reduced. Dead ends are on a website having poor linking structure.

The web structure of service pages may have pages with a dead end. The end causes no further flows for further action and no internal links. Good website structures have the pages designed such that they specifically gently guide the visitors toward actions and towards next step. For example, if one searches for a book title on Amazon, then visitor gets links of other books also on a similar topic.

9.4.2.2 Analyzing and Implementing a System with Web Graph Mining

Number of metrics analyze a system using web graph mining. Following are the examples:

1. In-degrees and out-degrees
2. Closeness is centrality metric. Closeness, $C_c(v) = 1 / \sum_{u \in V} gdist(v, u)$, where $gdist$ is the geodesic distance between vertex v with u and sum is over all u linked with V . Geodesic distance means the number of edges in a shortest path connecting two vertices. Assume v has an edge with w , and w has an edge with u . Assume u does not have direct edge from v . Then, geodesic distance = 2 (two edges between v and u in shortest path).
3. Betweenness
4. PageRank and LineRank
5. Hubs and authorities
6. Communities parameters, triangle count, clustering coefficient, K-neighbourhood
7. Top K-shortest paths

9.4.3 Computation of PageRank and PageRank Iteration

Assume that a web graph models the web pages. Page hyperlinks are the property of the graph node (vertex). Assume a Page, Pg (v) in-links from Pg (u), and Pg (u) out-linking similar to Pg (v), to total $N_{out}[(Pg(u)]$ pages. Figure 9.9 shows Pg (v) in-links from Pg (u) and other pages.

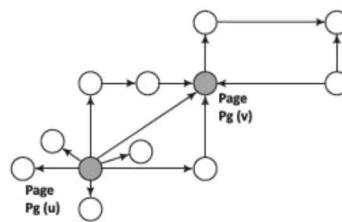


Figure 9.9 Page Pg (v) in-links from Pg (u) and other pages

N_{out} for page U is 7 and for V is 1 in the figure. Number of in-linking N_{in} for page V is 4. Two algorithms to compute page rank are as follows:

1. PageRank algorithm using the in-degrees as conferring authority

Assume that the page U, when out-linking to Page V "considers" an equal fraction of its authority to all the pages it points to, such as Pg v. The following equation gives the initially suggested page rank, PR (based on in-degrees) of a page Pg v:

$$PR(Pgv) = nc \cdot \sum_{Pgu: Pgu \rightarrow Pgv} [PR(Pgu)/N(Pgu)] \quad (9.21)$$

where $N(Pgu)$ is the total number of out-links from U . Sum is over all Pgv in-links. Normalization constant denotes by nc , such that PR of all pages sums equal to 1.

However, just measuring the in-degree does not account for the authority of the source of a link. Rank is flowing among the multiple sets of the links. When Pgv in-links to a page Pgu , its rank increases and when page Pgu out-links to other new links, it means that $N(Pgu)$ increases, then rank $PR(Pgv)$ sinks (decreases). Eventually, the $PR(Pgv)$ converges to a value.

Therefore, rank computation algorithm iterates the rank flowing computations as shown below:

EXAMPLE 9.7

Assume S corresponds to a set of pages. Initialize $\forall Pg \in S$. Symbols mean that initialize all pages Pg contained in the S and initialize Page Rank (Pgv) for each page as follows:

$$PRinit(Pgv) = 1/|S| \quad (9.22)$$

How are the page ranks of the pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means until converge)?

SOLUTION

Iterate and compute $PR(Pgv)$ for each page as follows:

Until ranks do not change (within specified margin) (that means converge)

{

for each $Pgv \in S$ compute,

$$PR(Pgv) = \sum_{Pgu: Pgu \rightarrow Pgv} [PR(Pgu)/N(Pgu)] \quad (9.22)$$

and normalization constant,

$$nc = \sum_{Pg \in S} [PRnew(Pgv)] \quad (9.23a)$$

$$\text{for each } Pgv \in S: PR(Pgv) = nc \cdot PR(Pgv) \quad (9.23b)$$

}

2. PageRank algorithm using the relative authority of the parents over linked children

A method of PageRank considers the entire web in place of local neighbourhood of the pages and considers the relative authority of the parents (children). The algorithm uses the relative authority of the parents (children) and adds a rank for each page from a rank source.

The PageRank method considers assigning weight according to the rank of the parents. Page rank is proportional to the weight of the parent and inversely proportional to the out-links of the parent.

Assume that (i) Page v (Pgv) has in-links with parent Page u (Pgu) and other pages in set $PA(v)$ of parent pages to v that means $\in PA(v)$, (ii) $R(v)$ is PageRank of Pgv , (iii) $R(u)$ is weight (importance/rank) of Pgu , and (iv) $ch(u)$ is weight of child (out-links) of Pgu . Then the following equation gives PageRank $R(v)$ of link v :

$$R(v) = \sum_{u \in PA(v)} [R(u)/|ch(u)|] \quad (9.25)$$

where $PA(v)$ is a set of links who are parents (in-links) of link v . Sum is over all parents of v . nc is normalization constant whose sum of weights is 1.

Assume that a rank source E exists that is addition to the rank of each page $R(v)$ by a fixed rank value $E(v)$ for Pgv . $E(v)$ is fraction α of $[1/|PA(v)|]$.

An alternative equation is as follows:

$$R(v) = nc \cdot \left\{ (1-\alpha) \sum_{u \in PA(v)} \left[\frac{R(u)}{|ch(u)|} \right] + \alpha \cdot E(v) \right\}. \quad (9.26)$$

where $nc = [1/R(v)]$. $R(v)$ is iterated and computed for each parent in the set $PA(v)$ till new value of $R(v)$ does not change within the defined margin, say 0.001 in the succeeding iterations.

Significance of a PageRank can be seen as modeling a “random surfer” that starts on a random page and then at each point: $E(v)$ models the probability that a random link jumps (surfs) and connect with out-link to Pgv. $R(v)$ models the probability that the random link connects (surf) to Pgv at any given time. The addition of $E(v)$ solves the problem of Pgv by chance out-linking to a link with dead end (no outgoing links).

Therefore, rank computation algorithm iterates the rank flowing computations as shown in Example 9.8.

EXAMPLE 9.8

Assume PA corresponds to a set of parent pages to a page v. Initialize $\forall Pg \in PA(v)$. Symbols mean that initialize all pages u contained in the set of parent pages of PA(v) and initialize Page Rank R(v) for each page as follows:

$$R(v) = [1/|PA(v)|]$$

How are the page ranks of the pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means until converges)?

SOLUTION

Iterate and compute R(v) for each page as follows:

Until ranks do not change that means converges (within specified margin, say 0.001)

for each $v \in PA(v)$ compute,

$$R(v) = nc \cdot \left\{ (1-\alpha) \sum_{u \in PA(v)} \left[\frac{R(u)}{|ch(u)|} \right] + \alpha \cdot E(v) \right\} \quad (9.27)$$

and normalization constant,

$$nc = \sum_{u \in PA(v)} [R(u)] \quad (9.28a)$$

$$\text{for each } v \in PA(v): R(v) = nc \cdot R(v) \quad (9.28b)$$

PageRank Iteration using MapReduce functions in Spark Graph

The computation of PageRank using SparkGraph method (Section 8.5),

```
graph.pageRank(0.0001).vertices
ranksByUsername = users.join(ranks).map{case id, (username, rank)) => (username, rank)}
```

The method includes conversions to MapReduce functions and using HDFS compatible files. Functions PageRank(), ranksByUsername() do the computations using the PageRankObject. GraphX consists of these functions (GraphOps). GraphX Operators includes the functions (Section 8.5).

Static PageRank algorithm runs for a fixed number of iterations, while dynamic PageRank runs until the computed rank converges. Convergence means that after certain iterations, the rank does not change significantly and any change remains within a pre-specified tolerance. Thereafter the iterations stop.

Assume specified tolerance at the start of iterations is 0.0001 (1 in 10000). When the rank does not change beyond that tolerance, it means rank value will converge and then the iterative process will stop.

9.4.4 Topic Sensitive PageRank and Link Spam

Number of methods have been suggested for computations of topic-sensitive page ranking, R_{TS} . The $R_{TS}(v)$ of a page P(v) may be higher for a specific topic compared to other topics. A topic associates with a distinct bag of words for which the page has higher probability of surfing than other bags for that topic.

Topic-sensitive PageRank method uses surfing weights (probabilities) for the pages containing the topic or bag of words corresponding to a topic. Method for creating topic-sensitive PageRank is to compute the bias to rank $R(v)$ and thus increase the effect of certain pages containing that topic or bag of words.

Refer equation (9.25) for computations of $R(v)$, and equation (9.26) for computations after introducing additional influence to the page. A method of introducing biasing is simple. It assumes that a rank source E exists that is additional having in-links from other pages, and thus adds to the rank of each page $R(v)$ by a fixed (uniform) or non-uniform weight factor α . The factor α is a multiplication factor to actual in-links without the bias.

Recapitulate equation (9.26). Probability of random jump to page v is $E(v)$. An alternative equation for topic sensitive PageRank, $R(v)$ computation for page $P(v)$ is as follows:

$$R(v) = nc \cdot \left\{ (1 - \alpha_t) \cdot P(v) \sum_{u \in PA(v)} \left[\frac{R(u)}{\lceil ch(u) \rceil} \right] + \alpha_t \cdot E(v) \right\}. \quad (9.29)$$

Probability of random jump to page v is $E(v)$. $\alpha_t = 0$ for page unrelated to a topic α is not 0 for page related to a topic. α_t = surfing probability for in-links for a topic t . Further, coefficient $(1 - \alpha)$ is considered as biasing factor depending on the web page $P(v)$ selected for a queried topic t .

The page is having in-links from other pages. Assume N_t is number of topics to which a page is sensitive to surfing those topics. Effect of topics on PageRanks increases by using a non-uniform $N_t \times 1$ personalization vector for surfing probability p . Higher α means higher p .

Assume that the topics are t_1, t_2, \dots, t_n . Fix the number N_t . R_{TS} is to be computed for each of them. Therefore, compute for each topic t_j , the PageRank scores of page v as a function of t_j , which means compute $R(v, j)$, where $j = 1, 2, \dots, n$. That also means compute the n elements of a non-uniform $N_t \times 1$ personalization vector $R_{TS}(v)$ for t_1, t_2, \dots, t_n .

Link Spam

Effects of a *link spam* can be nullified using the topic-sensitive PageRank algorithm. Link Spam tries to mislead the PageRank algorithm. A link spam attempts to make PageRank algorithm ineffective. The spam assisting pages connects to the page repeatedly and increases the in-degree of a page, thereby enhancing the rank to a large value.

A link spam creator website w_s also has a page l_s for whom w_s attempts to enhance the PageRank. The w_s has a large number of assisting pages al_s which out-links to l_s only. The al_s pages also prevent the PageRank of l_s from being lost. A spam mass consists of w_s , l_s and its al_s pages.

Methods nullify the effect by introducing a trust rank for a page u used in equation (9.29) and tracing spam mass of in-link pages to the page v .

Following are the steps for finding spam mass:

1. A distant topic sensitive page has unusually high in-degrees compared to the other pages of the same topic. A plot known as power-law plot is drawn between the log of number of web pages on the y-axis out-linking to the page v and logs of them in-degrees of v on the x-axis.
2. Plot is nearly linear as the number exponential decays is within degrees. N is proportional to $\exp(-d)$, where d is decay constant.
3. An unusual pattern with marked deviation from near linearity identifies the distant link spam mass.

9.4.5 Hubs and Authorities

A hub is an index page that out-links to a number of content pages. A content page is topic authority. An authority is a page that has recognition due to its useful, reliable and significant information.

Figure 9.10(a) shows hubs (shaded circles) with the number of out-links associated with each hub. Figure 9.10(b) shows authorities (dotted circles) with the number of in-links and out-links associated with each link.

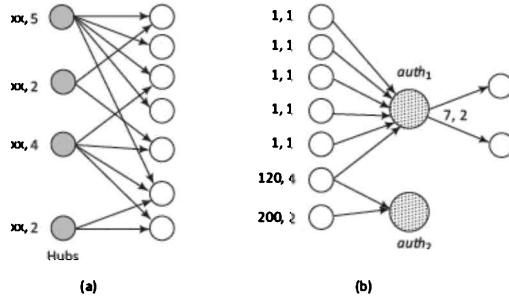


Figure 9.10 (a) Hubs (shaded circles) and (b) Authorities (dotted circles)

In-degrees (number of in-edges from other vertices) can be one of the measures for the authority. However, in-degrees do not distinguish between an in-link from a greater authority or lesser authority.

Authority, $auth_1$, in Figure 9.10(b) has in-links from 6 vertices (in-degrees = 6) and $auth_2$ has in-links to just 2 (in-degree = 2). However, $auth_1$ has link with six vertices with in-degrees = 1, 1, 1, 1, 1 and 120 (total = 125). Authority, $auth_2$ has links with two vertices with in-degrees = 120 and 200 (total = 220). $auth_2$ has association with greater authorities. Therefore, in-degrees may not be a good measure as compared to authority.

Kleinberg (1998) developed the Hypertext-Induced Topic Selection (HITS) algorithm.⁶ The algorithm computes the hubs and authorities on a specific topic t . The HITS analyses a sub-graph of web, which is relevant to t . Basis of computation is (i) hubs are the ones, which out-link to number of authorities, and
(ii) authorities are the ones, which in-link to number of hubs. A bipartite graph exists for the hubs and authorities.

Consider a specifically queried topic t . Following are the steps:

1. Let a set of pages discover a root set R using standard search engine. Root pages may limit to top 200 for t .
2. Find a sub-graph of pages S , using a query that provides relevant pages for t and pointed by pages at R . Sub-graph S pages form Set for computations as it includes the children of parent R and limit to a random set of maximum 50 pages returned by a “reverse link” query.
3. Eliminate purely navigational links and links between two pages on the same host.
4. Consider only u ($\|u\| \approx 4-8$) pages from a given hyperlink as pointer to any individual page. (Section 9.4.2)

Sub-graph for HITS consisting of root set R of pages and children of parents in the sub-graph S . Figure 9.11 shows subgraph S for HITS consisting of root set R of pages and all the pages pointed to by any page of R .

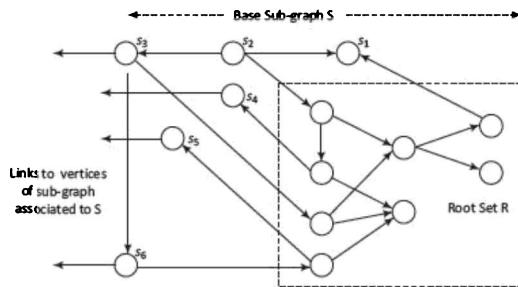


Figure 9.11 Sub-graph for HITS consisting of root set R of pages and base sub-graph S including all the pages pointed to by any page of R .

The left directed leftmost arrows from s_3 , s_4 , s_5 and s_6 are pointing to nodes in sub-graph(s) associated to S . The following example explains the algorithm steps to compute hub score and authority score.

EXAMPLE 9.9

Assume that v has number of in-links and v has number of out-links. Assume S corresponds to base set of pages and R corresponds to root set. (i) Initialize S to R . (ii) Initialize $\forall u \in S$. Symbols mean that initialize all pages u , contained in the S .

(iii) Normalization constant is nc. The (i) hub (v) hub score and (ii) auth authority score of page v for each page is as follows:

For each $v \in S$, $\text{auth}(v) = 1$; $\text{hub}(v) = 1$; $nc = 1$; (9.30)

How are the hub and authority of pages in a given set of pages iterated and computed till the ranks do not change (within specified margin, that means until converges)? Usually 20 iterations converge the result within margin, usually set to 0.001.

SOLUTION

Iterate and compute auth (v) and hub (v) for each page as follows:

Until ranks do not change (within specified margin) (that means converges)

{

for each $v \in S$ compute,

$$\text{auth}(v) = nc \cdot \sum_{u: u \rightarrow v} [\text{hub}(u)], \quad (9.31a)$$

$$\text{hub}(v) = nc2 \cdot \sum_{u: v \rightarrow u} [\text{auth}(u)], \quad (9.31b)$$

and normalization constant,

$$nc1 = \sum_{p: v \in S} 1 / [\text{auth}(v)]^2. \quad (9.32a)$$

$$nc2 = \sum_{p: v \in S} 1 / [\text{hub}(v)]^2. \quad (9.32b)$$

$$\text{for each } v \in S: \text{auth}(v) = nc1 \cdot \text{auth}(v); \text{auth}(v) = nc2 \cdot \text{auth}(v); \text{auth}(v); \quad (9.32c)$$

}

Difference between HITS and PageRank

HITS considers mutual reinforcement between authority and hub pages. PageRank ranks the pages just by authority and does not take into account distinctions between hubs and authorities. HITS considers the local neighbourhood between 4 to 8 pages surrounding the results of a query, whereas PageRank is applied to the entire web. HITS depends on topic t, while PageRank is topic-independent. PageRank effects by dead-ends.

9.4.6 Web Communities

Web communities are web sites or collections of websites, which limit the contents view and links to members. Examples of web communities are social networks, such as LinkedIn, SlideShare, Twitter and Facebook.

The communities consist of sites for do-it-yourself sites, social networks, blogs or bulletin boards. The issues are privacy and reliability of information.

Metric for analysis of web-community sites are web graph parameters, such as triangle count, clustering coefficient and K-neighbourhood.

K-neighbourhood analysis means the number of 1st neighbour nodes, 2nd neighbour nodes, and so on ($K = 1, 2, 3, 4$ and so on).

K-core analysis means the number of cores within a marked area. A core may consist of a triangle of connected vertices. A core may consist of a rectangle with interconnected edges and diagonals. A core may also be a group of cores.

Spark GraphX (Section 8.5) described functions for degree centralities, degree distribution, separation of degree, betweenness centralities, closeness centralities, neighbourhoods, strongly connected components, triangle counts, PageRank, shortest path, Breadth First Search (BFS), minimum spanning tree (forest), spectral clustering and cluster coefficient.

9.4.7 Limitations of Link, Rank and Web Graph Analysis

Following are the limitations of link and web graph analysis:

1. Search engines rely on metatags or metadata of the documents. That enhances the rank if metadata has biased information.
2. Search engines themselves may introduce bias while ranking the pages of clients higher as the pages of advertising companies

may provide higher searches and hence lead to biased ranks.

3. A top authority may be a hub of pages on a different topic resulting in increased rank of the authority page.
4. Topic drift and content evolution can affect the rank. Off-topic pages may return the authorities.
5. Mutually reinforcing affiliates or affiliated pages/sites can enhance each other's rank and authorities.
6. The ranks may be unstable as adding additional nodes may have greater influence in rank changes.

Self-Assessment Exercise linked to LO 9.3

1. Write and explain the equations for computing PageRank using relative authority of parent nodes.
2. Show diagrammatically network organization model of directed graphs for the structure of the web. How are the page hub and page authority computed?
3. What are the metrics which Spark GraphX compute?
4. How does the equation for computing the hub of a page differ from the computing authority of a page?
5. How does link spam function? How is the link spam discovered from the plot between the number of web pages and in-degrees?

9.5 SOCIAL NETWORKS AS GRAPHS AND SOCIAL NETWORK ANALYTICS

A **social network** is a social structure made of individuals (or organizations) called “nodes,” which are tied (connected) by one or more specific types of inter-dependency, such as friendship, kinship, financial exchange, dislike or relationships of beliefs, knowledge or prestige. (*Wikipedia*)

Social networking is the grouping of individuals into specific groups, like small rural communities or some other neighbourhoods based on a requirement. The following subsections describe social networks as graph, uses, characteristics and metrics.

LO 9.4

Representation of social networks as graphs, methods of social network analysis, finding the clustering in social network graphs, evaluating the SimRank, counting triangles (cliques) and discovering the communities

9.5.1 Social Network as Graphs

Social network as graphs provide a number of metrics for analysis. The metrics enable the application of the graphs in a number of fields. Network topological analysis tools compute the degree, closeness, betweenness, egonet, K-neighbourhood, top-K shortest paths, PageRank, clustering, SimRank, connected components, K-cores, triangle count, graph matches and clustering coefficient. Bipartite weighted graph matching does collaborative filtering.

Apache Spark GraphX and IBM System G Graph Analytics tools are the tools for social network analysis.

Centralities, Ranking and Anomaly Detection

Important metrics are degree (centrality), closeness (centrality), betweenness (centrality) and eigen vector (centrality). Eigen vector consists of elements such as status, rank and other properties. Social graph-network analytics discovers the degree of interactions, closeness, betweenness, ranks, probabilities, beliefs and potentials.

Social network analysis of closeness and sparseness enables detection of abnormality in persons. Abnormality is found from properties of vertices and edges in network graph. Analysis enables summarization and find attributes for *anomaly*.

Social network characteristics from observations in the organizations are as follows:

1. Three-step neighbourhoods show positive correlation between a person and high performance. Betweenness between vertices and bridges between numbers of structures are not helpful to the organization. Too many strong links of a person may have a negative correlation with the performance.
2. Social network of a person shows high performance outcome when the network exhibits structural diversity. Person with a social network with an abundant number of structural holes exhibits higher performance. This is because having diverse relations help an organization.

Social network analysis enables detection of an anomaly. An example is detection of one dominant edge which other sub-graphs are follow (succeed). *Ego network* is another example. The network structure is such that a given vertex corresponds to a sub-graph where only its adjacent neighbours and their mutual links are included.

The analysis enables spam detection. Spam is discovered by observation of a near star structure. Figure 9.12 shows discovering anomaly, ego-net and spam from the analysis.

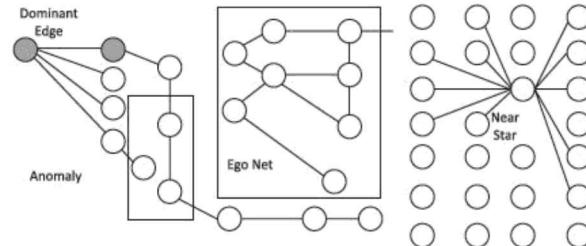


Figure 9.12 Discovering anomaly, ego-net and spam (using near star) from the analysis

Social network has concerns of privacy, security and falsehood dissimilation. Security issues are phishing attacks and malwares.

9.5.2 Social Graph Network Topological Analysis using Centralities and PageRank

Social graph network can be topologically analyzed. The centralities (degree, closeness, effective closeness and betweenness) and PageRank (vertexRank similar to PageRank in web graph network) are the parameters analyzed.

Degree

Degree of a graph vertex means the total number of edges linked to that. *In-degree of a vertex* means the number of in-edges from the other vertices. *Out-degree of a vertex* means the number of out-edges to other vertices to which that vertex directs. *Degree distribution function* means the distribution function for the degrees of vertices (Section 6.2.5 described the common distribution functions).

Closeness

Graph vertex closeness $c_c(v)$ is a way of defining the centrality of a vertex in reference to other vertices. Sum is the overall vertices connected to other vertices u . The u is a subset of vertices in set V .

The centrality (closeness index), c is function of distances of vertices.

$$c_c(v) = \left[\sum_{u \in V} d(u, v) \right]^{-1}.$$

where $d(u, v)$ is distance between u and v for path traversal.

Effective Closeness

Effective closeness $C_{ec}(v)$ can also be analyzed. Use approximate average distance from v to all other vertices in place of the shortest paths. C_{ec} reduces run time for cases with a large number of edges and near linear scalability in computations.

Betweenness

Graph vertices betweenness means the number of times a vertex exists between the shortest path and the extent to which a vertex is located ‘between’ other pairs of vertices. Betweenness $c_b(v)$ of a vertex v requires calculating the lengths of shortest paths among all pairs of vertices and computations of the summation for each pairing vertex in V .

PageRank

PageRank is a metric for the importance of each vertex in a graph, assuming an edge from v_1 to v_2 represents endorsement of importance of v_2 by v_1 by connecting, following, interacting, opting for relationship, sharing belief or some other means.

Contacts Size

Contacts size means a vertex connection to many vertices. The size of each vertex does not convey any meaningful information. A big social graph network will also require high maintenance cost.

Indirect Contacts

Indirect contacts metric means betweenness, which is the sum of the shortest paths within geodesic distances from all other pairing vertices. Three-step contact metric means a number of edges to other vertices plus the number of edges from other vertices within geodesic distances = < 3.

Both metrics convey meaningful information. The indirect contacts metric has meaning in terms of magnitude of betweenness centrality.

Structure Diversity

Structure diversity metric means that social graph has access to diverse sub-graphs (knowledge).

9.5.3 Social Graph Network Analysis using K-core and Neighbourhood Metrics

K-core is a sub-graph in a graph network structure. *Graph Vertex Kth neighbourhood* is number of 1st neighbour vertices, 2nd neighbour vertices and so on to a querying vertex that are correlated, linked, and have weighted correlations or the associations.

K-nearest neighbourhood (KNN) finds *K*-similar objects, items, or entities, which are nearest neighbours after computing the similarities. For example, KNN is *K*-documents (or books) in the large number of text documents (books) that are most similar to the queried document.

Collaborative filtering for frequent itemsets uses weighted bipartite graph matching.

Figure 9.13 shows the *K*-cores and *K*-neighbourhoods metrics for a social network graph. The figure also shows frequent itemsets obtained from collaborative filtering algorithm (Sections 6.4 and 6.8.1).

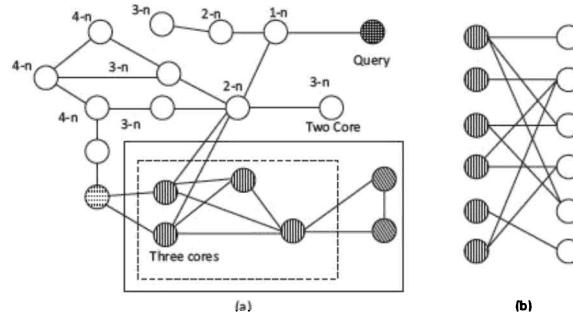


Figure 9.13 (a) *K*-cores and *K*-neighbourhoods with *K* = 1, 2, 3 and 4 and (b) Frequent itemsets from collaborative filtering algorithm (weighted bipartite graph matching)

Figure 9.13(a) shows three cores of two triangles, one quadrilateral, two cores of one pentagon and one triangle in *K*-neighbourhoods. *K* = 1, 2, 3 and 4. Figure 9.13(b) shows frequent itemsets from collaborative filtering algorithm (weighted bipartite graph matching).

9.5.4 Clustering in Social Network Graphs

One of the methods of detecting communities from social graph analysis is finding clustering and cluster coefficients. A clustering coefficient is a metric for the likelihood that two associated vertices of a vertex are also associated with other vertices. A higher clustering coefficient indicates a greater association and cohesiveness.

Connected components mean components of the datasets (represented by properties of vertices) connected together. For example, finding student-teacher datasets, social network datasets, etc.

9.5.5 SimRank

Similarity can be defined by properties of graph vertices. For example course, subject, student, scientist, Java programmer, status, values, or any other salient characteristic. Social network analysis of graphs computes *SimRank*.

SimRank is the metric for measuring similarity between vertices of the same type. The computation starts from a vertex possessing specific property and path traversals through the edges search the similarities. The vertices having similar properties are counted to the *SimRank*. The counting continues till counts per unit traversals converge within a prefixed margin, say .001. *SimRank* converges to a value which is applicable for path traversals within, say geodesic distance, say up to 200. The computations are analogous to ones for *PageRank* as in Example 9.7

9.5.6 Counting Triangles and Graph Matches

One of the methods of detecting communities is counting of triangles. A triangle means three vertices forming a triangle with edges interconnecting them.

Triangle count refers to the number of triangles passing through each vertex. The count is a measure of clustering. A vertex is part of a triangle when it has two adjacent vertices with an edge between them.

Graph matches are computed using filtering or search algorithm, which uses the properties, labels of vertices, edges or the geographic locations.

Figure 9.14 shows triangles and triangles between similar graph properties found from graph matches. Edge labels show the GPAs of students socially connected.

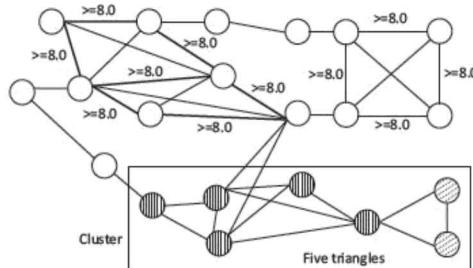


Figure 9.14 Clustering of five triangles and three matches of graphs

9.5.7 Using SparkGraph (Map-Reduce) for Network Graphs

Section 8.5 describes Spark GraphX algorithms for analyzing graphs. Connected components compute by `graph.connectedComponents().vertices` method in `SparkGraph`. Connected Components Algorithm labels each connected component of the graph with an ID. Each connected component ID is ID of the lowest-numbered vertex. For example, in a social network, connected component objects can approximate clusters. GraphX contains an implementation of the algorithm in the `ConnectedComponentsObject`. The clusters are found by discovering close-by connected components using closeness centrality metric.

SparkGraphX triangle-count algorithm computes the number of triangles passing through each vertex. The count is a measure of clustering. `TriangleCount` requires the edges to be in canonical orientation (`srcId < dstId`). Source vertex ID is `srcId` and Destination vertex ID is `dstId`. Graph is partitioned using `Graph.partitionBy` operator.

9.5.8 Direct Discovery of Communities

Three metrics identify groups and communities from a social graph:

1. Cliques – A clique forms by a set of vertices when each of the vertices directly connects to every other individual vertex through the edges. Detecting the cliques leads to direct discovery of communities.
2. Structurally cohesive blocks.
3. Social circles from connections and neighbourhoods

A bridge enables the link between two groups. Application of analyzing communities, SimRanks and bridges are finding a set of experts, specific areas of expertise, and ranking the expertise in an organization.

Experience in social science fields shows that the social network of a person is the key indicator of the stature of the person and his/her success potential. Social graph analysis enables finding key bridges and persons with most connections.

Figure 9.15 shows a social graph with two cliques and a bridge.

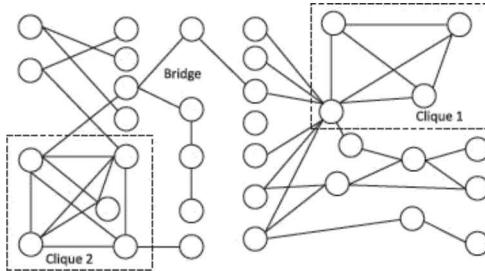


Figure 9.15 Two cliques in a social graph network and a bridge between the cliques

Clique 1 has set of four vertices, each connected with three edges to three others. Clique 2 has five vertices, each connected by edges to other four. Two edges in the figure provide the bridge between two sub graphs, on left and right sides.

Self-Assessment Exercise linked to LO 9.4

1. How do the metrics analyze a social network graph of persons in an organization? How do they relate to inter-dependency, performance, groups, expertises, beliefs, knowledge or prestige?
2. Define the terms degree, closeness, betweenness, egonet, K-neighbourhood, top-K shortest paths, PageRank, clustering, SimRank, connected components, K-cores, triangle count, graph matches and clustering coefficient.
3. How the cliques discover communities from social network analysis?
4. What are the uses of Apache GraphX ConnectedComponents and triangles count methods in social graph analysis?

KEY CONCEPTS

anomaly detection
 authority
 bag of words
 betweenness
 centralities
 clique
 closeness
 collaborative filtering
 collection wide-word frequency
 concept extraction
 content relevance
 document frequency
 documents classification
 documents clustering
 effective closeness
 ego net
 feature selection
 HITS algorithm
 hub
 hyperplane

in-degrees
KNN
link analysis
marginalization
Naïve Bayes classifier
out-degrees
outliers
PageRank
part-of-speech tagging
pattern analysis
pattern discovery
sequential patterns
SimRank
social network
social network graph
spam detection
structured text
SVM classifier
term frequency
text analytics process pipeline
text cleanup
text features generation
text mining
text pre-processing
TF-IDF
Top K shortest paths
triangles count
unstructured text
vector space model
web community
web content analytics
web graph
web structure
web usage mining

Learning Outcomes

LO 9.1

1. Text mining techniques help revealing the patterns and relationships in large volumes of textual content that are not directly

- visible. The mining leads to new business opportunities and improvements in processes.
2. Five phases in text mining are (i) text pre-processing, (ii) feature generation, (iii) feature selection, (iv) text data mining, and (v) analysing the results. Text analytics involves provisions of strong integration with the already existing database, artificial intelligence, machine learning, and text mining techniques such as, information retrieval, natural language processing, classification, clustering and knowledge management, respectively.
 3. Machine learning based text classification methods are (i) K nearest neighbour classifier, (ii) Naïve Bayes method, (iii) decision trees, (iv) decision rules classification, and (v) support vector machines.
 4. Naïve Bayes classifier is a simple, probabilistic and statistical classifier. The classifier computes the conditional probability tables.
 5. SVMs based classifier is a discriminative classifier formally defined by a separating hyperplane. SVM seeks a decision surface to separate the training data points into two classes and makes decisions based on the support vectors that select the only effective elements in the training set.

LO 9.2

1. Web data mining is a process of discovering patterns in large datasets to gain knowledge. The process can be shown as Raw Data → Patterns → Knowledge. Web data refers to (i) web content – text, image, records, (ii) web structure – hyperlinks and tags, and (iii) web usage – http logs and application server logs.
2. Steps for pre-processing of web-data are quite similar to pre-processing for text mining. The steps include collection of wide-word frequencies and document frequencies. Machine learning techniques for web content analytics are clustering, classification and association rule mining.
3. Web usage mining discovers and analyses the patterns in click stream and associated data generation and collection as a consequence of user interactions with web resources on the World Wide Web.
4. A link spam creator website *ws* also has a page *ls*. *ws* attempts to enhance the PageRank of *ls*.
5. A hub is an index page that out-links to number of content pages. A content page is topic authority. Authority is a page that has recognition due to provisioning useful, reliable and significant information. HITS algorithm computes the hubs and authorities on a specific topic *t*.
6. Web community is website or collection of the websites that limits the view of contents, and that links the members (for example, LinkedIn). Metric for analysis of web community sites are web graph parameters, such as triangle count, clustering coefficient and K-neighbourhood.

LO 9.3

1. Link analysis enables finding the PageRank, centralities, hubs, and authorities. Page ranking method considers the entire web in place of local neighbourhoods of the pages. PageRank of a page refers to relative authority of the parents out-linking to the page.
2. Web structure models as directed-graphs network-organization. A page may have a number of hyperlinks directed by out-edges to other pages. Text at the hyperlink represents the property of vertex that describes the destination of the out-going edge. A hyperlink in-between the pages represents the conferring of the authority.
3. SparkGraph includes conversions to MapReduce functions and use HDFS compatible files. Page rank() and ranksByUsername() are static and dynamic methods compute on the PageRankObject

LO 9.4

1. A social network is a social structure made of individuals (or organizations) called “nodes”, which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, financial exchange, dislike or relationships of beliefs, knowledge or prestige
2. Social network topological analysis tools compute the degree, closeness, betweenness, egonet, K-neighbourhood, Top-K shortest paths, PageRank, clustering, SimRank, connected components, K-cores, triangle count, graph matches and clustering coefficient. Bipartite weighted graph matching does collaborative filtering.
3. Analysis enables summarization and find attributes for anomaly.
4. Apache Spark GraphX includes PageRank, ConnectedComponents and TrianglesCount algorithms, and fundamental operations

for social graph analytics.

5. Analysis of cliques discovers groups and communities. Analysis also finds the bridge between the cliques.

Objective Type Questions

Select one correct-answer option for each questions below:

9.1 The term *text analytics* evolves from (i) provisioning of strong integration with the already existing (ii) database, (iii) artificial intelligence, (iv) machine learning, and (v) text Data Store techniques such as (vi) information retrieval, (vii) natural language processing, (viii) classification, (ix) clustering, and (x) knowledge management, respectively.

- (a) all except ii and iv
- (b) ii to ix
- (c) all except ii, iii, iv and vii
- (d) all

9.2 SVMs main uses are (i) classification based on the outputs taking discrete values in a set of possible categories, (ii) separation or prediction, if something belongs to a particular class or category. Other uses are (iii) finding a decision boundary between two categories, (iv) clustering, (v) regression analysis, and regression, if continuous real-valued output (continuous values of x_1 , in place discrete n values, x_2, x_3, \dots, x_n), and (vi) discriminative classifier.

- (a) all except iii
- (b) i, ii and iv
- (c) all except iv
- (d) i to v

9.3 Applications of (i) web content mining, and (ii) web-structure mining of web documents are: (iii) Classifying the web documents into categories, (iv) identifying the topics of the web documents, (v) creation of tables and databases, (vi) finding similar web pages across different web servers, and (vii) relevance or the applicability of web content measured with respect to a basis, such as making recommendations, filtering or querying

- (a) all except vii
- (b) all except ii
- (c) all except iv
- (d) i to v

9.4 The HITS analyses a (i) subgraph of web, which is relevant to (ii) topic t, (iii) query q. The assumptions are (iv) authorities are the ones, which out-link to number of hubs, and (v) hubs are the ones, which in-link to number of authorities. (vi) A bipartite graph exists for the hubs and authorities. (vii) First set of pages discovers a root set R using standard search engine, then (viii) finds a sub-graph of pages S, using a query that provides relevant pages for t and pointed by pages at R.

- (a) all except iii to v
- (b) all except iii and vi
- (c) all except vi
- (d) all except iv and vi

9.5 Web contents mining tasks are: (i) finding clustering, (ii) classifying, (iii) mining association rules, and (iii) topic identification, tracking and drift analysis for adding new documents to a collection library. Other tasks are: (iv) assigning by rechecking for the emergence of new topics, (v) creation of concept hierarchy, building of category dimensions, such as domain, location, time, application, privileges, and (vi) measuring the relevance or the applicability of web content on basis of documents, queries, roles or tasks or user profiling.

- (a) all except vii and viii
- (b) all

- (c) all except vii
- (d) All except vi to viii

9.6 The most common form of pattern analysis consists of (i) a knowledge query mechanism such as SQL, (ii) loading usage data into a data cube in order to perform OLAP operations,

(iii) visualization techniques, such as graphing patterns or assigning colors to different values. The analysis also finds (iv) content and structure information which can filter out patterns containing pages of a certain usage type, content type, or pages that match a certain hyperlink structure.

- (a) all except iii
- (b) all
- (c) all except i and ii
- (d) i to ii

9.7 PageRank method considers (i) the entire web in place of a local neighbourhood of the pages, (ii) queries top 10 pages, and (iii) considers the relative authority of the children pages with respect to parent page. *PageRank method* considers assigning weight (iv) as 1, and (v) according to the rank of the parents. (vi) PageRank is inversely proportional to the weight of the parent and proportional to out-links of the parent.

- (a) i, iii, and v
- (b) i and v
- (c) all except ii and vi
- (d) all

9.8 Social network graph analysis tools do the (i) clustering analysis which means the number of 1st neighbour nodes, 2nd neighbour nodes, and so on. ($K = 1, 2, 3, 4$ and so on), (ii) social network community and network analysis. The graph analysis finds the (iii) close-by entities, (iv) fully mesh-like connected sets, (v) network graph analysis beside centralities, (vi) also does computations of the *property* of the links, (vii) rectangle counts, and (viii) clustering coefficient.

- (a) i to vi
- (b) all except i, vi and vii
- (c) ii to iv
- (d) i to iii, v, viii

Review Questions

9.1 How are the features evaluated in the text documents? **(LO 9.1)**

9.2 Explain five phases and steps in the phases during text analytics. **(LO 9.1)**

9.3 When is the Naïve Bayes conditional probabilities based classifier used? When is the support vectors based discriminative-classifier used? Write details of each. **(LO 9.1)**

9.4 What are the tasks in web data analytics? Describe the pre-processing steps and mining tasks in web contents analytics. **(LO 9.2)**

9.5 How is the emergence of new topics discovered? How do concept hierarchy create and build from category dimensions, such as domain, location, time, application and privileges? **(LO 9.2)**

9.6 How does the web usage mining discover and analyze patterns in click stream, and generate and collect associated data? **(LO 9.2)**

9.7 Describe various link analysis metrics used for analytics. How is PageRank iterated and computed using relative authority of in-linking pages? How does ranking algorithm compute topic-sensitive PageRank? **(LO 9.3)**

9.8 How does structure of web model as graph network? Draw a diagram for web graph nodes and edges. What are the metrics computed for a web graph? **(LO 9.3)**

9.9 Describe HITS algorithm to iterate and compute the hubs and authorities? **(LO 9.3)**

- 9.10 How does social graph analysis relate to positivity and negativity analysis about the persons? How does social graph network anomaly detection help an organization? (LO 9.4)
- 9.11 How are social graph analytics metrics, degree, closeness, betweenness, egonet, K-neighbourhood, Top-K shortest paths and SimRank computed by path traversals? (LO 9.4)
- 9.12 What are the operators provisioned in Apache Spark GraphX for social network graphs analysis? (LO 9.4)

Practice Exercises

- 9.1 List the steps in the methods used for grouping the text documents into clusters, automating the document organization, topic extraction. Take the example of HTML pages or your University or Company website. (LO 9.1)
- 9.2 Explain how text analytics tasks performs using Python library *nltk*. (LO 9.1)
- 9.3 List the steps in document clustering method. How do you use the clusters for the fast information retrieval or filtering? Take the example of student grade cards or Company annual reports. (LO 9.1)
- 9.4 List the steps in classifying web documents into categories, identifying similar pages across different web documents to classify them as web pages of a university or company. (LO 9.2)
- 9.5 List the steps in recommendations for top N relevant documents in a collection or portion of a collection. List the steps in filtering – show/hide documents based on most/least relevancy.(LO 9.2)
- 9.6 Using Example 9.7, write algorithms for PopularityRank, SimRank and best student search. (LO 9.3)
- 9.7 Rewrite PageRank and HITS algorithms using vectors and matrices. (LO 9.3)
- 9.8 Write the steps in performing bipartite weighted graph matching in social network graph analysis. (LO 9.4)
- 9.9 Describe steps to compute the triangles, junction trees, shortest paths and top K-shortest paths and discover the communities in social network graphs of students. (LO 9.4)

1 http://www.nactem.ac.uk/brochure/NaCTeM_Brochure.pdf

2 https://www.ibm.com/support/knowledgecenter/en/SS3RA7_18.1.1/ta_guide_ddita/textmining/shared_entities/tm_intro_tm_defined.htm

3 <https://blogs.aws.amazon.com/bigdata/post/Tx22THFQ9MI86F9/Applying-Machine-Learning-to-Text-Mining-with-Amazon-S3-and-RapidMiner>

4 https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

5 <http://papers.cumincad.org/data/works/att/2873.content.pdf> “The Anatomy of a Large-Scale Hypertextual Web Search Engine” Sergey Brin Lawrence Page, 1998

6 J. Kleinberg (1998), Authoritative sources in a hyperlinked environment, Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms. A longer version appears in the Journal of the ACM 46, 1999. Available from http://www.cis.hut.fi/Opinnot/T-61.6020/2008/pagerank_hits.pdf

Note:

○○● Level 1 & Level 2 category

○●● Level 3 & Level 4 category

●●● Level 5 & Level 6 category

Chapter 10

Programming Examples in Analytics and Machine Learning using Hadoop, Spark and Python

LEARNING OBJECTIVES

After studying this chapter, you will be able to:

- LO 10.1 Learn steps for installation of Hadoop and Spark, and storage and processing of Big Data and large datasets
- LO 10.2 Get acquainted with steps of programming for deploying and exploring the open source Lego datasets and its schema, processing and storage of datasets, counting of dataset items using MapReduce, creating HBase tables from the CSV format datasets and creating DataFrames from the RDDs
- LO 10.3 Get knowledge of programming steps in Hive and PySpark for operations of Merge and Join on DataFrames, usages of SQL-equivalent functions for join and processing queries, and using the UDFs
- LO 10.4 Get introduced to data visualization using Python Plotting (graphing) library, Programming for pi graph, bar charts and scatter plots
- LO 10.5 Learn steps in the programs and machine learning algorithms for the clustering, classification, regression analysis and predictive analytics

Readers shall also be able to learn the following from the ***Practice Exercises*** given at the end of the chapter:

1. Big Data analytics applications in business
2. Uses of datasets for analyzing and predicting future
3. Develop codes to solve problems in analytics, predict and visualize using Sklearn, PySpark and Mahout

RECALL FROM EARLIER CHAPTERS

Previous chapters 1 to 9 described the followings:

- Hadoop, MapReduce, map tasks using key-value pairs, Hadoop ecosystem tools, Hbase, Hive, HiveQL (DDL, DML, Querying data, Aggregation and Join), Pig (Pig Latin data model, commands, relational operations, Eval functions and user defined functions), Apache Spark
- Spark for distributed and faster in-memory analytics, cluster computing and APIs in Java, Scala, Python and R
- Spark architectural features, software stack components, their functions, steps in data analysis with Spark, ETL processes using built-in functions and operators, ETL pipelines, analytics, data/information reporting, visualizing methods, using Spark with Python advanced features, and UDFs, vectorized UDFs, grouped vectorized UDFs
- Developing and testing Spark codes, programming with RDDs, applications of MLlib, and Machine Learning (ML) methods for analysis of datasets
- Apache Mahout architecture, components, their applications for clustering analysis, classification, Naïve Bayes analysis, SVMs for analytics, collaborative filtering, recommender system, and regression analysis for predictions
- Stream computing and SparkStreaming
- Graph databases, graph analytics, Apache SparkGraphX, its Architecture, components, and their applications for graph analytics
- Text mining, web content and web usage analytics, link analysis and web structure analytics

This chapter focuses on Hadoop/Spark/PySpark programming examples. Programs explore datasets, perform analytics and demonstrate machine learning algorithms and data visualization.

10.1 INTRODUCTION

Hadoop provides Big Data storage and computing using clusters. Hadoop manages both large-sized structured and unstructured data in different formats efficiently and effectively. The formats, such as XML, CSV, JSON, text files. Hadoop ecosystem provides running of applications on Big Data. Hadoop deploys MapReduce, HBase distributed databases and other application programming models. Hadoop ecosystem includes Hive and Pig. Hadoop applications support layer and application layer components include Hive, Pig Spark, Spark and Mahout.

Apache® Spark™ is an advanced Big Data analytics tool. Spark is fast and general compute engine. Spark provides in-memory, distributed and faster cluster computing. Spark supports data stored at HDFS, Hadoop compatible data source, such as HBase, Cassandra, Ceph and Amazon S3.

Spark SQL includes SQLContext and JDBC datasource that can read from (and write to) SQL databases. Spark SQL provides DataFrames(SchemaRDDs) to allow processing of a large amount of structured data. Spark SQL does the following: runs SQL-like scripts for query processing, using catalyst optimizer and tungsten *execution engine*, processes structured data, and provides flexible APIs for support for many types of data sources.

SparkSQL has built-in functions and operators for creating ETL pipelines and analytics. Spark SQL does ETL operations by creating ETL pipeline on the data from different file-formats, such as JSON, Parquet, Hive, Cassandra and then running ad hoc queries.

Spark enables programming with the RDDs, and machine learning applications with MLlib. Apache Mahout and components have applications for the development of clustering, classification, collaborative filtering and recommender system algorithms.

Spark provides APIs in Java, Scala, Python and R. Spark architecture has many features, and software stack components for data analysis. Apache Spark contains interactive shell for Python programming known as PySpark. PySpark embodies the advanced features of Python, such as UDFs, vectorized UDFs and grouped vectorized UDFs. Python has strong libraries for analytics, machine learning and natural language processing.

Spark Streaming is a stream-processing platform for data mining and real-time analytics. Stream processing requires samples of streaming data, and does the filtering, counting of distinct elements, analysis of frequent itemsets, and mining of association rules. The analysis gives the count of the instances of frequent itemsets present in the stream. Spark streaming facilitates real-time sentiment analytics and stock prices analytics.

Spark is thus one of the most important components for Big Data Analytics Stack. This chapter focuses on Hadoop/Spark/PySpark program examples. The programs explore datasets, perform analytics, visualize data, and run machine learning algorithms using famous toy company, Lego Inc. open source datasets.

Section 10.2 describes installation methods for Hadoop, Hive, Pig and Spark on Ubuntu platform.

Section 10.3 describes datasets used in the examples in subsequent sections of this chapter. The section describes deploying and exploring Lego datasets, schema, processing and storage, MapReduce implementation for counting items in the dataset, creating Hive data tables from CSV format dataset and creating DataFrame from the RDDs.

Section 10.4 describes Hive and PySpark programs using functions, Merge and Join of DataFrames, SQL equivalent join functions and UDFs for customized query processing.

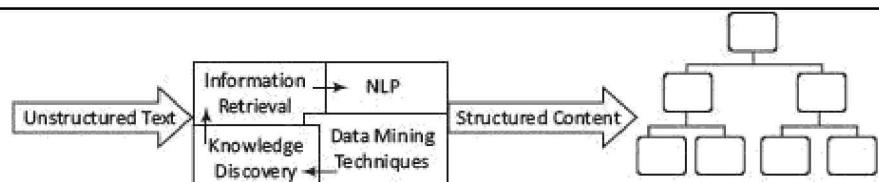
Section 10.5 describes programs for data visualization using pi, bar and scatter plots.

Section 10.6 describes machine learning programs using sklearn for SVM, Naïve Bayes classifiers, linear and polynomial regression analysis and predictive analytics.

Practice exercises at the end of the chapter describe the csv files of open source datasets of an automobile company. Datasets for new car sales can be used for analyzing and predicting future sales. The datasets contain monthly car sales for 2007–2017 by the *make* and the data for the most popular car models. The exercises for the analytics shall make a reader through in understanding of algorithms described in the Chapters 5 and 6, and the usage of PySpark and Mahout. Online solution-guide associated with the book explains the methods and codes for them.

10.2 | INSTALLATION STEPS FOR HADOOP AND SPARK

The following subsections describe the steps for installation of Hadoop and Spark processes, and configuration of platform used for computing.



10.2.1 Installation Steps for Hadoop, Hive and Pig

Following are the steps for Hadoop Installation for setting up of a single-node cluster of Hadoop 2.9.0 on Ubuntu 16.04 Operating System.¹ Latest version is Hadoop 3.0²

which is in alpha phase. Apache community has incorporated many changes and is still working on some of them.

1. Updates all repositories

```
sudo apt-get update
```

2. Install Java and ssh

Hadoop Java Versions

Version 2.7 and later of Apache Hadoop requires Java 7. Earlier versions (2.6 and earlier) support Java 6.

```
sudo apt-get install openjdk-7-jdk
```

```
sudo apt-get install ssh
```

3. Download Hadoop-2.9.0.tar.gz, hive-1.2.2.tar.gz, pig-0.17.0 files.

The Apache™ Hadoop® project and other Hadoop-related projects at Apache are available at: <http://hadoop.apache.org>

4. Create a dedicated user hduser. This creates a directory in “home” name as “hduser”.

5. Copy the given hadoop-2.9.0.tar.gz, hive-1.2.2.tar.gz, pig-0.17.0 files into the “hduser” directory

6. Extract all files in the “hduser” directory:

```
tar -xvzf hadoop-2.9.0.tar.gz
```

7. Go to hadoop-2.9.0/conf/

Following XML files are present:

- 1) core-site.xml
- 2) hdfs-site.xml
- 3) mapred-site.xml
- 4) yarn-site.xml

- 7.1 Open core-site.xml with text editor. Copy the following lines in to core-site.xml:

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:8020</value>
    </property>
</configuration>
```

- 7.2 Open hdfs-site.xml with text editor. Copy the following lines in to hdfs-site.xml:

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/home/hduser/hadoopdata/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/home/hduser/hadoopdata/hdfs/datanode</value>
    </property>
</configuration>
```

- 7.3 Open yarn-site.xml with text editor. Copy the following lines in to yarn-site.xml:

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>MapReduce shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
</configuration>
```

- 7.4 Open mapred-site.xml with text editor. Copy the following lines in to mapred-site.xml:

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
```

- 7.5 Open hadoop-env.sh with text editor

Copy the following line where the JAVA_HOME path is given in to hadoop-env.sh

or below this line “# export JAVA_HOME=/usr/lib/******/”:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk
```

8. Update \$HOME/.bashrc

open bashrc file with command:

```
sudo gedit ~/.bashrc
```

copy following lines to the End of File:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk
export HADOOP_HOME=/home/hduser/hadoop-2.9.0
export HIVE_HOME=/home/hduser/hive-1.2.2
export PIG_HOME=/home/hduser/pig-0.17.1
export PATH=$PATH: $JAVA_HOME/bin: $HADOOP_HOME/bin:
$HIVE_HOME/bin: $PIG_HOME/bin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

9. Refresh the bashrc file

```
source ~/.bashrc
```

10. Disable the SSH

```
ssh localhost (enter the password)
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

11. Format the namenode

```
hadoop namenode -format  
or  
hadoop namenode -format -force
```

12. Start hadoop

```
start-all.sh
```

13. Check all services of Hadoop are running or not using JPS (Java Virtual Machine Process Status Tool) command.

```
jps
```

Check whether the following processes are running:

```
7770 RunJar  
8704 JobTracker  
8929 Jps  
6495 RunJar  
8316 NameNode  
8610 SecondaryNameNode  
8460 DataNode  
8866 TaskTracker
```

The installation is successful if all the above processes are displayed on console.

10.2.2 Installation Steps for the Spark on Ubuntu

Section 5.4 introduced steps for downloading Apache Spark, getting started, Spark shell, developing and testing Spark codes, programming with the RDDs and applications to MLlib. The steps for installation of Spark on Ubuntu are as follows:

Step 1: Installing Java

Check whether Java is already installed.

```
java -version
```

Step 2: Install Scala

```
sudo apt-get install scala
```

Check whether Scala installed.

```
scala  
Test scala.  
println("Hello World")  
Quit Scala  
: q
```

Step 3: Install Spark

Download a pre-built for Hadoop 2.7 version of Spark (preferably, Spark 2.0 or later) from

<https://spark.apache.org/downloads.html>

Save .tgz file on computer.

Go to terminal and change directory to where .tgz file saved (or just move the file to r home folder), then use

```
tar xvf spark-2.2.1-bin-hadoop2.7.tgz
```

Extract the Spark folder and use.

```
cd spark-2.0.2-bin-hadoop2.7.tgz
```

```
mv spark-2.0.2-bin-hadoop2.7 spark
```

Make an entry for spark in .bashrc file

Edit the Hadoop user profile /home/hadoop/.profile and add the following lines:

```
export SPARK_HOME=/home/hadoop/spark
```

```
export PATH=$SPARK_HOME/bin: $PATH
```

Source the changed .bashrc file by the command

```
source ~/.bashrc
```

Then use

```
cd bin
```

and then

```
./spark-shell
```

Spark shell will pop up. Here, one can load

```
.scala scripts
```

Test the environment.

```
print(sc.version)
```

The output should read:

```
2.2.1
```

The `sc` is `SparkContext` that is automatically created for you when `PySpark` starts. Initializing a `PySpark` session creates `sqlContext` as well. To exit the `PySpark` session:

```
quit()
```

10.2.3 Computing Platform Configuration

The examples presented in this chapter are executed on a system with the following configuration:

- Operating System Ubuntu 16.04 LTS 64-bit
- Processor: Intel® Core™ i7-4790 CPU @ 3.60GHz x 8
- RAM: 16 GB
- Hard Disk: 116.5 GB
- Graphics: GeForce GT 710/PCIe/SSE2

10.3 DATASETS USED IN THE EXAMPLES, DATA DEPLOYMENT AND EXPLORATION

Analytics requires data. A collection of data is called a Dataset. A dataset must be strongly-typed, immutable collection of objects that map to a relational schema. A rich dataset is needed to provide high-quality analytics results. A dataset needs to be rich, which means must offer vast opportunity for exploration and offer an immense range of data patterns.

Here, we have chosen datasets from Lego database. Lego is a renowned brand of construction-toys. The different varieties of toys are sold in sets which build a specific object.

LO 10.2

Steps of programming for deploying and exploring the open source Lego datasets, specifying the schema, the processing and storage of datasets, counting of dataset items using MapReduce, creating HBase tables from the CSV format datasets and creating DataFrames from the RDDs

A set contains colourful interlocking plastic bricks and other parts. The number, sizes and shapes of the parts vary with sets. The parts can be assembled and connected in many ways, to construct objects, vehicles, buildings and working robots. The parts can be taken apart to make other objects as well.

The Lego database contains information about the parts in different Lego sets. This dataset was compiled by *Rebrickable*³ for kaggle.com as a public dataset [LDB⁴], with the objective of using these files for any purpose.

This dataset contains Lego sets from year 1950 to July, 2017. The dataset is arranged in 8 CSV files.

The dataset comprises various toy themes. There are in all 614 themes defined in themes.csv for various Lego toys, such as Robot, Airport, Building and Train. Almost 11,673 sets are manufactured in year starting from 1950 to July, 2017 (sets.csv). Sets have different number of parts (field: num_parts in sets.csv).

Table 10.1 shows a sample row of sets.csv that represents a Tractor set with set number 378-1 developed in 1972. It belongs to Theme Id 397 and has 36 parts in total:

Table 10.1 A sample row of sets.csv file

set_num	Name	Year	theme_Id	num_parts
378-1	Tractor	1972	397	36

The inventories of parts and sets are also provided in the dataset. Inventory Ids of 11,681 parts and their set numbers are available in inventories.csv. Overall there are 25,993 parts (parts.csv) belonging to 57 different part categories (Part_categories.csv), whereas inventory of parts contains 5,80,251 parts (inventory_parts.csv). Inventory of parts also stores the colour information of the part (field: color_id in inventory_parts.csv). Similarly, the inventory of sets contains 2,846 sets (inventory_sets.csv).

The colours are defined in colors.csv where the RGB values of 133 colors and two values for unknown color and No color are specified. The colors are also featured as transparent and non-transparent using Boolean value is_trans. Sample of colors.csv is presented in Table 10.2.

Table 10.2 Sample of colors.csv file

Id	Name	rgb	is_trans
-1	Unknown	0033B2	f
4	Red	C91A09	f
36	Trans-Red	C91A09	t

f stands for false and **t** for true.

Figure 10.1 shows the dataset schema of the Lego database that helps in figuring out how the files are related.

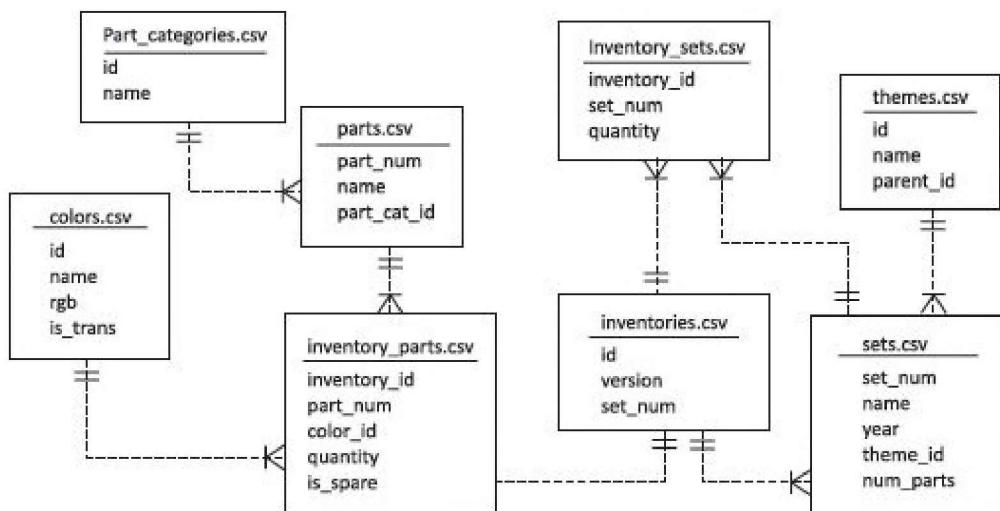


Figure 10.1 Schema of Lego Database [LDB]

10.3.1 Counting and Sorting of Items in Datasets using MapReduce

Objective of Program MapReduce program presented in the section counts sets of toys in the given dataset on the basis of year of introduction and sorts them from the highest count to the lowest.

Mapper loads the input csv file (`sets.csv`), prepares dataframe with two columns (year and number of parts) and passes the prepared dataframe to the reducer.

Reducer loads the dataframe obtained from the Mapper, performs an aggregate function, `count`, to count the year wise sets, and sorts the sets in descending order of count.

Mapper and Reducer programs use the dataframe to store the data extracted from files. Section 5.3.1 describes the concept of Dataframe and Section 10.3.3 describes the method of creating DataFrame From CSV file and the RDD.

Input file(s): `sets.csv`

Code of Mapper

```
#file: mapper.py

import pandas as pd #import pandas python package
#Define mapper Function
def mapper():
    #Extract records of CSV file into Dataframe df1
    df1 = pd.read_csv('/home/.../sets.csv')

    #Prepare dataframe with two columns(year and number of
    #parts)only
    df2 = df1[["year","num_parts"]].copy()

    # Now print out the data that will be passed to the
    #reducer
    print(df2.to_string(index=False))

#Execute mapper function
mapper()
```

Run the Mapper Code using command on the prompt

\$: python mapper.py

Output of Mapper (Note: Shown only 10 records, Total there are 11,673 records):

year	num_parts
1970	471
1978	12
1987	2
1979	12
1979	12
1979	12
1979	18

```
1978      15
1976      147
1976      149
```

Note: The output of mapper.py transfers to reducer.py in MapReduce.

Code of Reducer

```
#file: reducer.py
import pandas as pd #import pandas python package
import sys           #import System-specific parameters
                     #and function module
import re            #import regular expressions module
                     #for splitting the console input data

def reducer():
    #obtain the list of lines from stdin
    data = sys.stdin.read().splitlines()
    #split each line into 2 items: 'year' and 'num_parts'
    data = [re.split(r'\s+(?=\\d+$)', l) for l in data]

    #construct the dataframe
    df = pd.DataFrame(data, columns=['year', 'num_parts'])
    #Count the rows Group by year and arrange them in
    #ascending order
    sd=df.groupby('year')
    ['year'].count().sort_values(ascending=False)
    print(sd) #Display desired output

#Execute reducer function
reducer()
```

Note: Regular expressions \s matches Unicode whitespace characters and \d matches any Unicode decimal digit. ? =... is *lookahead assertion*. The pattern character '\$' matches at the end of the string.

Run the Reducer Code using command on the prompt

```
$: python mapper.py | python reducer.py
```

The above command executes the `mapper.py` file and the output of `mapper.py` is provided to `reducer.py`.

(Note: The symbol “|” denotes a pipe. The Pipe is a command in Linux that allows to use two or more commands such that output of one command serves as input to the next.)

Output of Reducer (Counts sets on the basis of year of introduction and sorts them from the highest count to the lowest):

year	
2014	713
2015	665
2012	615
2016	596
2013	593
2011	503
2002	447
2010	444
2003	415
2009	402
2004	371
2008	349
2001	339
2005	330

10.3.2 Storing CSV Dataset into Hive Database

Create a Table in Hive

```
hive> create table sets(set_num string, name string,  
year int, theme_id int, num_parts int, category string)  
row format delimited fields terminated by ',';
```

Output of Create Table Query

OK

Time taken: 0.107 seconds

Load the Data

```
hive> Load data local inpath '/home/... /sets.csv'  
overwrite into table sets;
```

Output of Load Data Query

Loading data to table default.sets

Table default.sets stats: [numFiles =1, numRows =0, totalSize = 507514, rawDataSize = 0]

OK

Time taken: 1.124 seconds

Initial step in this exploration process is to read in the data and print a quick summary statistics.

10.3.3 Storing CSV Dataset into the Spark DataFrame

A Spark DataFrame is a two-dimensional data structure with rows and columns. It is similar to a matrix of data rows or a table in relational database or an Excel sheet with column headers. Columns may contain data of different types.

A sample of one of the data file is as follows:

```
set_num, name, year, theme_id, num_parts  
00-1, Weetabix Castle, 1970, 414, 471  
0011-2, Town Mini-Figures, 1978, 84, 12  
0011-3, Castle 2 for 1 Bonus Offer, 1987, 199, 2
```

The file format is csv (comma separated values). Each row of the data is a different record of set (toy set), and different data fields within each row are separated by commas. The first row is the header row and illustrates each data field. Remaining rows contains the data values for data fields correspondingly. The entire set of one data field (say set_num) of all the rows, is a column. The dataset can be visualized in matrix format as:

set_num	name	year	theme_id	num_parts
00-1	Weetabix Castle	1970	414	471
0011-2	Town Mini-Figures	1978	84	12
0011-3	Castle 2 for 1 Bonus Offer	1987	199	2

Reading the data Reading the data requires creation of a dataframe first. This can be done in multiple ways:

- Using different data formats. For example, loading the data from JSON, CSV.
- Loading data from Existing RDD (Resilient Distributed Dataset).
- Programmatically specifying schema

Create a DataFrame from CSV file Pandas is a high level data processing and analysis library of Python, which provides easy-to-use data structures. Pandas is built on functionality provided by the Numpy package and its key data structure is the DataFrame.

Import pandas for data processing functions with `import pandas as pd`. The data from a CSV file is used to create DataFrame, using the `read_csv` method.

Following code creates a dataframe, `toy_sets`, from csv file, `sets.csv` and print the column name of dataframe:

```
# Import the pandas library.
import pandas as pd
# Read in the data.
toy_sets = pd.read_csv("sets.csv")
# Print the names of the columns in toy_sets.
print(toy_sets.columns)
```

Output The code above reads the data in, and shows all the column names:

```
Index(['set_num', 'name', 'year', 'theme_id', 'num_parts'],
      dtype='object')
```

The shape of the data displays the number of rows and columns in the data file.

```
print(toy_sets.shape)
```

Output The toy_sets has 11673 rows, or sets, and 5 columns or data points describing each set:

```
(11673, 5)
```

10.3.4 Creating DataFrame from the RDD

First create a `SparkContext` to connect with Apache Cluster. When operations are required to be executed in a cluster `SparkContext` is needed. `SparkContext` specifies to Spark how and where to access a cluster. The `SparkContext` already exists when Spark shell is used. Otherwise, it can be created by importing, initializing and providing the configuration settings:

```
from pyspark import SparkContext  
sc = SparkContext()
```

Two ways to prepare a DataFrame from RDD are as follows:

1. Wrap the elements that belong to the same row in DataFrame by a parenthesis at the time of creation of RDD by `parallelize` function. Name the columns by `toDF` function where all the columns' names are wrapped by a square bracket.

```
rdd = sc.parallelize([(10, 20, 30), (11, 21, 31), (12, 22, 32)])  
dataFrame = rdd.toDF(["p", "q", "r"])
```

2. Use `pyspark.sql` and assign a name to each element in every row. Now convert the RDD into a DataFrame by `toDF` function in which there are no other names.

```
from pyspark.sql import Row  
rdd = sc.parallelize([Row(p=10, q=20, r=30), Row(p=11, q=21, r=31),  
Row(p=12, q=22, r=32)])  
df = rdd.toDF()
```

10.4 ! PROGRAMMING STEPS USING HIVE AND PYSPARK

The following subsections describe Merge and Join functions for DataFrame objects, analysis using UDFs for query-processing in Hive and Pyspark.

10.4.1 Merge and Join Functions for DataFrame Objects

Pandas provides a `merge` function as the entry point for all standard database join operations between DataFrame objects.

Syntax of merge function:

```
pd.merge(left,      right,      how='inner',
on=None, left_on =None, right_on = None, left_index = False,
right_index = False, sort = True)
```

Following are the meanings of the terms used:

1. *left* – First Dataframe object.
2. *right* – Second Dataframe object.
3. *how* – Similar to SQL join types ('left', 'right', 'outer' and 'inner'). *inner* is default value. (Table 10.3).
4. *on* – Columns (names) to join on. The column name should be present in both the left and right DataFrame objects. If value of *on* is None, the intersection of the columns in both DataFrames are selected to join by default.
5. *left_on* – Columns from the left Dataframe to use as keys.
6. *right_on* – Columns from the right Dataframe to use as keys.
7. *left_index* – Join keys from left Dataframe is set as the index, when set as true. In case of a Dataframe with a multiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.
8. *right_index* – Similar to *left_index* for the right Dataframe.
9. *sort* – Sort the result DataFrame by the join keys in lexicographical order. True is default value.

Table 10.3 Merge types and their SQL equivalent names

Merge options	SQL Equivalent	Description
left	LEFT OUTER JOIN	Use keys from left object
right	RIGHT OUTER JOIN	Use keys from right object
outer	FULL OUTER JOIN	Use union of keys
inner	INNER JOIN	Use intersection of keys

The Join operations on DataFrames are also available in Pandas. The operations are

like SQL JOIN in RDBMS. They are high performance in-memory join with full features. This implies Join columns with other DataFrame either on index or on a key column.

Syntax of join function

```
Dataframe.join(other, on = None, how='left', lsuffix = '',
rsuffix = '', sort = False)
```

Following are the meanings of the terms used:

1. Dataframe: First Dataframe object.
2. other – Second Dataframe object
3. on – Columns (column name, tuple/list of column names, or array-like) to join on. The column name should be present in both the left and right DataFrame objects.
4. how – Similar to SQL join types. Values are ‘left’, ‘right’, ‘outer’, ‘inner’. left is the default value (Table 10.3).
5. lsuffix – Suffix to use from left frame’s overlapping columns. Its data type is string.
6. rsuffix – Suffix to use from right frame’s overlapping columns. Its data type is string.
7. sort – Order result DataFrame lexicographically by the join key. If false, preserves the index order of the calling (left) DataFrame. False is the default value.

10.4.2 Analysis and Query-Processing Using UDFs in Hive and Pyspark

UDF is a customized function for which user specifies the code, input datasets and output datasets. A UDF helps in query processing for total inventories for user defined input itemsets and user defined outputs. Examples of inputs and outputs in the UDF are as follows:

1. Input Datasets (Color), (Year), (Color and Year), (Theme and Color), (Theme, Color and Year)
2. Output Datasets (Inventory), (Color), (Theme)

Python scripts for Customized UDF in Hive Hive provides for writing codes for the UDFs, similar to other programming languages. A UDF enables the code to introduce any new functions to the cluster for computations, as needed. Hive has limited built-in Hive functions (Section 4.4.7). Each dataset specifies a schema and has several features.

The analytics require additional ‘user defined functions’ (UDFs) over and above built-in functions.

UDF are implemented for actions, such as transformations and even for aggregations. User-Defined Aggregation Functions (UDAFs) transform a group of rows into one or more rows, meaning that one can reduce the number of input rows to a single output row by some customized aggregation function. An example of coding in HiveQL, that feeds the data to the Python script is given below. The code uses standard input and reads the result from its standard out.

Custom UDF in Python Objective: Return the toys category in text format, whether *new* or *intermediate* or *old* on the basis of year.

NumPy is the fundamental package for scientific computing with Python. Following code uses Python library numpy:

```
#myUdf.py
import sys
import numpy as np
import datetime
#Define udf (User defined function)
#Function returns category in text format on the basis of year
def year_to_rank(year):
    if year >= 2015: return 'New'
    elif year >= 2010: return 'Intermediate'
    elif year >= 1970: return 'Old'
    else: return 'Not known'
#input
for line in sys.stdin:
    line =line.strip()
    set_num,name,year,theme_id,num_parts,category=line.split('\t')
    name=name.lower()
    category=year_to_rank("year")
    print ('\t'.join([str(set_num),str(name),
                     year,theme_id,num_parts,str(category)]))
```

The above code is saved in a file named myUdf.py

Add Python script into Hive

It is essential to add the Python file as resource to the Hive cluster. The following command add file myUdf.py to the Hive cluster:

```
hive> add file /home/ ... /Lego/myUdf.py;
```

Output of add file command

```
Added resources: [/home/.../Lego/myUdf.py]
```

Run a Python UDF in Hive

```
hive> select transform(set_num,name,year,theme_id,num_parts,category)
using 'python myUdf.py' as set_num,name,year,theme_id,num_parts,category
from sets limit 10;
```

Output of transform Query

```
Query ID = hduser1_20180301115757_fa8bd58a-6e70-43ff-b8c3-f24f1d7e444c
```

```
Total jobs = 1
```

```
Launching Job 1 out of 1
```

```
Number of reduce tasks is set to 0 since there's no reduce operator
```

```
Job running in-process (local Hadoop)
```

```
2018-03-01 11:57:59,164 Stage-1 map = 100%, reduce = 0%
```

```
Ended Job = job_local1723749830_0002
```

```
MapReduce Jobs Launched:
```

```
Stage-Stage-1: HDFS Read: 642682 HDFS Write: 507514 SUCCESS
```

```
Total MapReduce CPU Time Spent: 0 msec
```

```
OK
```

set_num	name	NULL	NULL	NULL	New
00-1	weetabix castle	1970	414	471	old
0011-2	town mini-figures	1978	84	12	old
0011-3	castle 2 for 1 bonus offer	1987	199	2	old
0012-1	space mini-figures	1979	143	12	old
0013-1	space mini-figures	1979	143	12	old
0014-1	space mini-figures	1979	143	12	old
0015-1	space mini-figures	1979	143	18	old

0016-1	castle mini figures 1978 weetabix	186	15	old
00-2	promotional house 1	1976	413	147

Time taken: 1.256 seconds, Fetched: 10 row(s)

10.5 DATA VISUALIZATION USING PYTHON PLOTTING LIBRARY

Matplotlib is Python plotting library. Matplotlib is used to generate the output for the visualization. It could be interesting to plot a chart. The library imports using:

```
import pandas for data processing functions, and
import Matplotlib for visualizing the output.
```

Plotting functions in Matplotlib are included using import matplotlib.pyplot as plt statement. The functions in pyplot package facilitates drawing and showing the plots.

LO 10.4

Data visualization using Python Plotting (graphing) library and Programming for pi and bar charts and scatter plots

Pie Chart Plot Example

Objective Let's plot a pie chart of color transparency that can visualize the distribution of non-transparent and transparent colors.

Input file(s): colors.csv

Following is the code of plotting a pie chart:

```
import pandas as pd #Import pandas for processing the CSV
file I/O

import matplotlib.pyplot as plt #Import matplotlib
plotting library

#Read CSV file and Extract data into dataframe df
df = pd.read_csv('/home/.../colors.csv')

# Apply aggregation function count and store the result
in a new
```

```

# dataframe sd
sd = df.groupby('is_trans')['is_trans'].count

y = sd.values # Extract count values in array a

# Define an array, expl, which specifies the fraction of
# the
# radius with which to offset each wedge.
expl =(0.1, 0)

labels = 'Non-Transparent',                      # Define Data Labels
'Transparent'
colors = ['white', 'grey']                      # Define wedge colors

# Plot the pie chart
plt.pie(y, labels=labels, explode=expl, colors=colors,
autopct='%.1f%%', shadow=True, startangle=90)
plt.axis('equal')
plt.show()

```

The parameters of `pie` function are as follows:

1. *x*: Array of the wedge sizes.
2. *explode*: Array that specifies the fraction of the radius with which to offset each wedge.
3. *labels*: An optional list parameter with default value `None`. It is a sequence of strings providing the labels for each wedge.
4. *colors*: An optional array parameter with default value `None`. It is a sequence of `matplotlib` color arguments through which the pie chart will cycle. If `None`, will use the colors in the currently active cycle.
5. *autopct*: An optional parameter with default value `None`. It is a string or function that is a label of wedges with their numeric value. The label will be placed inside the wedge.

6. *shadow*: An optional Boolean parameter with default value False. It draws a shadow beneath the pie.
7. *startangle*: An optional float parameter with default value = None. If startangle not None, then rotates the start of the pie chart by an angle mentioned in degrees. The rotation is counterclockwise from the x-axis.
8. *radius*: An optional float parameter with default value None. It defines the radius of the pie. If radius is None it will be set to 1.

Figure 10.2 shows a pie chart output.

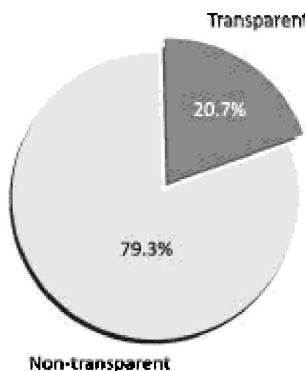


Figure 10.2 Pie chart depicting the color transparency

Example of Bar Chart Plot

Bar chart is another visualization tool. The chart presents the data of different groups that are being compared with each other. The plotting of bar chart requires data at x- and y-axes.

Objective Let's plot bar chart for number of parts in each category of toys.

Aggregation function `count` is used on `part_cat_id` field. The `part_cat_id` field is arranged in descending order before performing the aggregation.

Code for the bar chart is as follows:

```
import pandas as pd #Import pandas data processing,
import matplotlib.pyplot as plt #CSV file I/O functions
#Import matplotlib plotting library
```

```
#Read CSV file and Extract data into dataframe df
df = pd.read_csv('/home/.../parts.csv')

# Apply aggregation function count then sort function on
df

# and store the result in a new dataframe sd

sd = df.groupby('part_cat_id')
['part_cat_id'].count().sort_values(ascending=False)
y= sd.values # Extract count values
x =sd.index # Extract index
# Display the bar chart
plt.title('Bar Chart') #Define Title of the chart
plt.xlabel('Part Category Id') #Define Label of X-axis
plt.ylabel('Category Count')#Define Label of Y-axis
plt.bar(x, y,color = 'gray')# Plot the bar chart for x
and y
plt.show()# Display the bar chart
```

Input file(s) parts.csv

Figure 10.3 shows the bar chart depicting category-wise counts for parts.

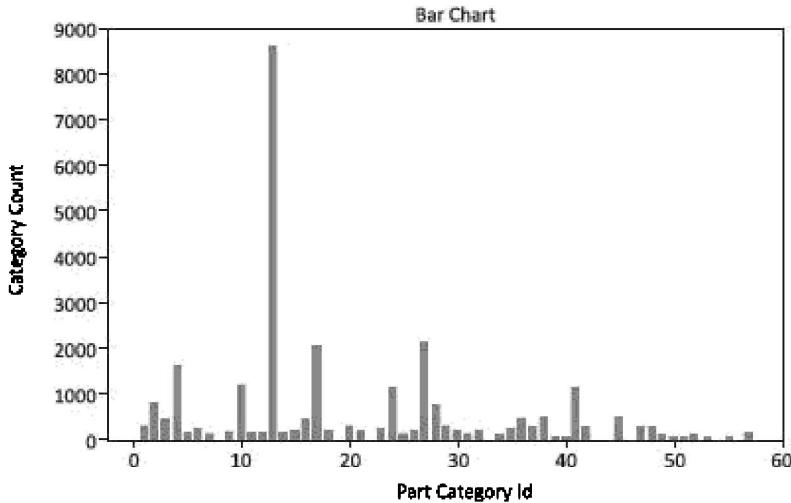


Figure 10.3 Bar chart depicting category-wise parts counts

The result in Figure 10.3 suggests that Category 13 has the highest number of parts (8,556), while Category 56 has the lowest number of parts (8).

Scatter Function Syntax

Scatter charts or scatter plots are used to plot data points on the horizontal and vertical axes. Scatter functions are used to draw scatter plot. Before an example of scatter plot, let us understand the various parameters of scatter function.

Syntax of scatter function is `(x, y, marker = "x", s = 150, linewidths = 5, zorder = 10)`

Following are the meanings of the terms used above:

1. `x, y`: Label or position, optional – Coordinates for each point.
2. `s`: Scalar or array_like, optional – Size of each point.
3. `color`: Label or position, optional, default: 'b' – Color of each point.
4. `marker`: MarkerStyle, optional, default: 'o' – Marker Symbol
5. `alpha`: Scalar, optional, default: None – The alpha blending value, between 0 (transparent) and 1 (opaque).
6. `linewidths`: Scalar or array_like, optional, default: None – The linewidth of the marker edges.
7. `Zorder`: Top-to-bottom order of the layers; Used when objects are overlapping in two-dimensional view

Table 10.4 shows eight common markers in plots.

Table 10.4 Eight common Markers

Marker	Symbol	Marker	Symbol
“.”	point	“v”	triangle_down
“,”	pixel	“^”	triangle_up
“+”	plus	“<”	triangle_left
“x”	cross	“>”	triangle_right

Now, let us consider two examples of scatter plots.

Example 1 of Scatter Plot

Objective Display a scatter plot on sets data of Lego database. This scatter plot displays the theme Id and number of parts defining the number of parts as datapoints in each theme.

Input file(s) sets.csv

```
#Import matplotlib plotting library
import matplotlib.pyplot as plt
# Import pandas for processing the CSV file I/O
import pandas as pd
#Read CSV file and Extract data into dataframe df
df = pd.read_csv('/home/ ... /sets.csv')
# Extract features from df and store them in arrays x and y
# respectively.
x = df['theme_id'].values #Extract theme_id values in array x
y = df['num_parts'].values #Extract num_parts values in array y
# Create Scatter plot of data points(x,y)
plt.scatter(x, y, color='gray', s=10) #s is size of scatter plot point
plt.xlabel('Theme Id')#Define Label of X-axis
plt.ylabel('Number of Parts') #Define Label of Y-axis
#plot the x & y axis# Define the min & max value of x axis and y axis
# as(x_min, x_max, y_min, y_max)
x_min = 0
```

```

x_max = 40
y_min = -1
y_max = 50
plt.axis([x_min, x_max, y_min, y_max])
plt.title('Scatter Plot') #Define title of the scatter plot
plt.show() #Display the scatter plot

```

Figure 10.4 shows a scatter plot which represents relationships between themes (represented by themeId) and number of parts.

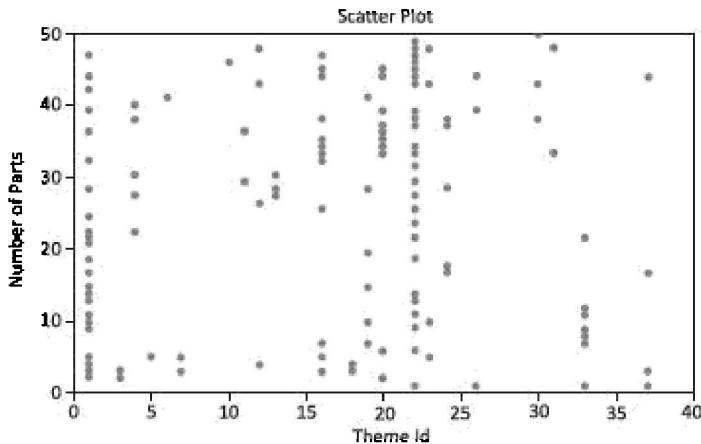


Figure 10.4 Scatter Plot which represents relationships between themes and number of parts

Example 2 of Scatter Plot

This example is for displaying a scatter plot for some random datapoints. numpy library is required for creating random number. `randint()` function present in this library returns random integers from the “discrete uniform” distribution in the “half-open” interval $(\text{low}, \text{high})$. If `high` is `None` (the default), then results are from $(0, \text{low})$. Size parameter of `randint` function defines the number of integers to be generated.

Objective Display a scatter plot on say, 50 random datapoints generated between $(1,1)$ and $(10,10)$.

Following is the code:

```

#Import matplotlib plotting library
import matplotlib.pyplot as plt

#Import NumPy for random number generation function

import numpy as np

#Generate 50 Random numbers(x, y) between (1,1) to
#(10,10)

x = np.random.randint(low=1, high=10, size=50)
y = np.random.randint(low=1, high=10, size=50)
#Create Scatter plot of data points (x,y)
plt.scatter(x, y,color='black',s=10) #s is size of point
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.axis([-1,11,-1,11])
plt.title('Scatter Plot')
plt.show()

```

Figure 10.5 shows a scatter Plot which depict randomly generated (x, y) data points. Output may vary if you execute the program again and again.

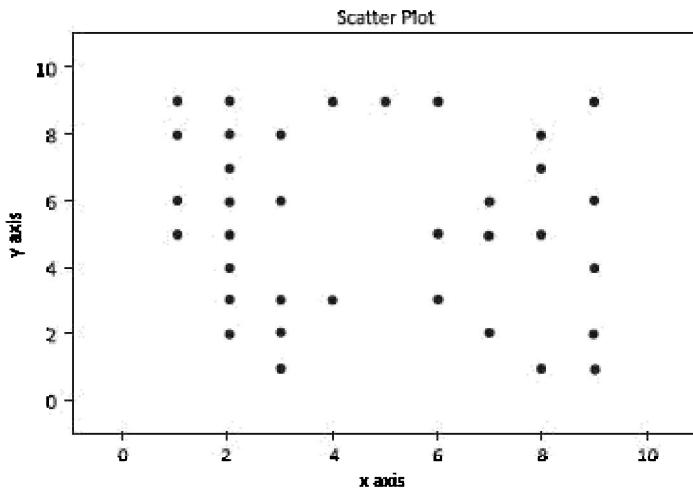


Figure 10.5 Scatter plot depicting randomly generated (x, y) data points

10.6 | MACHINE-LEARNING ALGORITHMS IMPLEMENTATION

Machine learning (ML) is a domain which focuses on the development of algorithms to exploit the data and learn from the data to make predictions. This has an immense range of applications, from effective web search to stock price prediction.

Python libraries Scikit-learn and Pandas are used to implement the ML algorithms. The library contains implementations of most common algorithms, including k-means, random forests, SVMs, and logistic regression. Scikit-learn has a consistent API for accessing these algorithms.

The following subsections provide insight into machine learning algorithms for clustering, classification and regression in Python.

10.6.1 Clustering Algorithm

Let us discuss a way to figure out more features about the datasets of Lego database. Clustering finds patterns within the data easily by grouping similar rows together.

Brief description of the K-means algorithm code is as follows:

1. Perform basic imports first. Import NumPy which is a scientific computing module (used here for returning a matrix from an array of year and color_id values, then matplotlib for graphing, and then K-means from sklearn).
2. Next, read the input csv files into individual dataframes and merge or join (use inner join) those to get a final dataframe with multiple attributes.
3. Initialize k-means with the required parameters to the K-means algorithm—(i) number of clusters (`n_clusters`) as 2. Change the `n_clusters` value to get output for n (say, 3 or 4) numbers of clusters. (ii) Another parameter is `random_state` which is an integer or None, used for initializing the internal random number generator, which decides initialization (of centroids). Remember, K-means is stochastic method (the results may vary even for the same input values). Hence, in order to make the results reproducible, one can specify a value for the `random_state` parameter.
4. Next, use `fit()` function to fit the data (learning)
5. Next, take the values found for the centroids, based on the fitment, as well as the labels for each centroid.
6. The “labels” here are labels that the machine assigns on its own. Similarly, the

Machine-learning programs using PySpark for the clusters identification, SVM and Naïve Bayes Classifiers and linear, and linear and polynomial regression analysis for enabling predictions

centroids also.

7. Finally, plot and visualize the machine's findings based on the data, and the fitment according to the number of clusters defined in the code.
8. x and y axes take the min_x, max_x, min_y and, max_y values. Here, theme Id is plotted on x-axis that has values 0 (minimum) to 614 (maximum). Color Id has values between 1 to 9999.
Restrict unknown (Color Id=-1), No Color (Color Id=9999) and New colors (Color Id from 1000 to 1007) for clearer visualization. User can implement the program for all colours to see the difference.
9. colors = [“g.”,”r.”,”c.”,”y.”] statement defines green, red, cyan and yellow colors for cluster no. 1, 2, 3 and 4, respectively.

Objective of the Program The cluster example prepares 2, 3 and 4 clusters of features theme_id and color_id. The program plots the resultant clusters using scatter plot. The clusters are represented in various colors with centroid drawn with X marker within each cluster. The values of centroids are also displayed.

Input file(s): sets.csv, inventories.csv, inventory_parts.csv and colors.csv

Following example implements K-means clustering algorithm and give full code:

```
#import the numpy, pandas, matplotlib and sklearn libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans #KMeans Clustering
Algorithm

#Read csv files and extract data in dataframes df1, df2,
df3 and df4
df1 = pd.read_csv('/home/.../sets.csv')
df2 = pd.read_csv('/home/.../inventories.csv')
df3 = pd.read_csv('/home/.../inventory_parts.csv')
df4 = pd.read_csv('/home/.../colors.csv')
#Restrict(unknown, No Color and New colors here for better
visualization.
df4 = df4[((df4['id'] <1000) & (df4['id'] >= 0))]
```

```
#Join/Merge all the dataframes into a new dataframe say,
df1 ##

df1 = pd.merge(df1, df2, left_on="set_num", right_on =
"set_num")
df1 = pd.merge(df1, df3, left_on="id", right_on =
"inventory_id")
df1 = pd.merge(df1, df4, left_on="color_id", right_on =
"id")

#Extract the theme_id and color_id features from merged
dataframe df1
#and Store in dataframe df
df = pd.DataFrame(df1, columns= ['theme_id', 'color_id'])

#Store feature data from df in the `f1` and `f2` arrays.
f1 = df['theme_id'].values #Extract theme_id column values
f2 = df['color_id'].values #Extract color_id column values

X=np.column_stack((f1, f2))

#Obtain minimum & maximum of X and y values for drawing
axes
min_x= f1.min()
max_x= f1.max()
min_y= f2.min()
max_y= f2.max()

#combine them into a feature matrix `X` before entering it
into the algorithm

#Execute KMeans algorithm
K=2 #Define number of clusters 2,3,4 here
kmeans = KMeans(n_clusters=K, random_state=1)
#fit the data
kmeans.fit(X)
```

```

#Obtain the values found for the Centroids & labels for
each centroid

labels = kmeans.labels_
centroids = kmeans.cluster_centers_
#Display the centroids and labels values before plotting
them

print("Centroids for ", K, "clusters are : ")
print(centroids)

print("Labels are : ")
print(labels)

#Define colours of the clusters. plot each cluster with a
#different colour. For example, 1 with green, 2 with red, 3
with Cyan,

#4 with yellow --- Add more colours for drawing more than 4
clusters

colors = ["g.","r.","c.","y."]

# plot the scatter graph for clusters

for i in range(len(X)):

plt.plot(X[i][0], X[i][1], colors[labels[i]],
markersize=10)

plt.scatter(f1, f2, colors="k")

plt.scatter(centroids[:, 0], centroids[:, 1], marker="x",
s=150, linewidths=5, zorder=10)

plt.axis([min_x, max_x, min_y, max_y])
plt.xlabel('Theme Id')
plt.ylabel('Color Id')
plt.show()

```

Output of the K-means algorithm Figure 10.6 shows K-means Output for 2, 3 and 4 clusters.

10.6.2 Classification Algorithm Example 1: SVM Classifier

SVM can be implemented with the help of scikit-learn library. Following is a simple demonstration which helps in building understanding of working with Linear Support Vector Classifier (SVC). The objective of a Linear SVC is to fit to the data provided and returning a “best fit” hyperplane that divides or categorizes the data.

Brief description of the SVM classifier algorithm code:

1. Perform basic imports first. Import NumPy is a scientific computing module (used here for returning a matrix from an array of year and theme_id features, then matplotlib for graphing, and then SVC from sklearn).
2. Next, Load the data from csv files into individual dataframes and join (use inner join) them to get a final dataframe of multiple attributes.
3. A function specifies Build_Data_Set using two parameters year and theme_id.
4. Next, fill the X parameter with the NumPy array containing rows of the above mentioned two features using np.array function. Then populate the y variable with the “targets” (or labels) converted to numerical data. Here, Boolean f is converted to 0 and t is converted to 1. The targets are basically binary or multivalued in a classification model.
5. The SVM calls the Build_Data_Set function, builds the linear SVC. Specify the kernel type to be used in the algorithm. Kernel type must be one of ‘linear’, ‘poly’, ‘rbf’ (Default), ‘sigmoid’, ‘precomputed’ or a callable.
6. Next, use fit () function to fit the predictor and target variables.
7. Calculate the feature weights and plot the scatter graph as well as classifier.
8. After being fitted, the model can also be used to predict new values using clfr.predict(X). For example, clfr.predict([2020]) to predict values in year 2020.

Objective of the Program The SVM classifier example which classifies the input dataset on the basis of transparency of the colors. Features selected are year and theme_id. The program plots the resultant classes using the scatter plot. The classes are represented using different colors. The dataset size is restricted for demonstration purpose only. User can execute the entire code on complete dataset. Here in the following example, we also restricted unknown (Color Id=-1), No Color (Color Id=9999) and New colors (Color Id from 1000 to 1007) for clearer visualization. User can implement the program for all colors to see the difference.

Input file(s) sets.csv, inventories.csv, inventory_parts.csv and colors.csv

Following is the complete code of SVM Classifier :

```
#import the numpy, pandas, matplotlib and sklearn libraries
import numpy as np
```

```

import pandas as pd
from matplotlib import pyplot as plt
from sklearn.svm import SVC #Support Vector Classifier

#Read csv file and extract data in dataframes df1, df2, df3
and df4
df1 = pd.read_csv("/home/.../sets.csv")
df2 = pd.read_csv("/home/.../inventories.csv")
df3 = pd.read_csv("/home/.../inventory_parts.csv")
df4 = pd.read_csv("/home/.../colors.csv")

#Join/Merge all the dataframes into a new dataframe say,
df##

df1 = pd.merge(df1, df2, left_on="set_num", right_on =
"set_num")
df1 = pd.merge(df1, df3, left_on="id", right_on =
"inventory_id")
df1 = pd.merge(df1, df4, left_on="color_id", right_on =
"id")
df1 = pd.DataFrame(df1, columns= ['year','theme_id',
'color_id','is_trans'])

#Define a function to build data set with two features
#The function returns the X(predictor) and Y(target)
def Build_Data_Set(features = ["year","theme_id"]):
    df = df1

    #Color Id=0, 9999 and 1000 to 1007 are ignored
    #for better visualization
    df = df[((df['id'] <1000) & (df['id'] >= 0))]

X = np.array(df[features].values)
Y = (df["is_trans"].replace("f",0).replace("t",1).values.tolist())

```

```

return X,y

#SVM Analysis Code begins from here
X, y = Build_Data_Set()
clf = SVC(kernel="linear", C= 1.0)
clf.fit(X,y)

#Calculate the feature weights
w = clf.coef_[0]
m = -w[0] / w[1]
xx = np.linspace(min(X[:, 0]), max(X[:, 0]))
yy = m * xx - clf.intercept_[0] / w[1]

#Plot the classifier
h0 = plt.plot(xx,yy, "k-", label="SVM Classifier")

#plot the scatter graph for classification
plt.scatter(X[:, 0], X[:, 1],c=y)
plt.axis([1950,2017,0,614])
plt.xlabel("Year")
plt.ylabel("Themes")
plt.legend()
plt.show()

```

Output of SVM classifier Figure 10.7 shows SVM outputs for two different data subsets

Salient Observations from SVM Outputs SVM does not directly provide probability estimates. The outputs illustrated in Figures 10.7(a) and (b) are not significant. When tried with the complete dataset (which has more than 5 lakh 70 thousand rows), the output is not revealing any significant interpretation. Taking the time in order of 1-2 hours for classifying complete dataset since single cluster machine is used here. The datasets have limited number of dimensions and the sample size is very large. It is identified that SVM does not perform well due to two reasons.

Reason 1: When the dataset is large, it requires high training-time thus, training becomes extremely slow.

Reason 2: When dataset has more noise, the target classes are overlapping.

Thus, from the practical experience, SVMs are proven to be better for small to medium datasets and datasets with low noise. SVMs are better for datasets with larger feature dimensions.^{5,6}

This also indicates that the suitability of the classifier depends upon the characteristics of datasets.

10.6.3 Classification Algorithm Example 2: Naïve Bayes Classifier

Naïve Bayes classification uses Bayes theorem to determine class of new data points. Consider an example where we want to know whether various themes and the colors used are more than 800 or not. The Naïve Bayes classification algorithm is presented in this section to classify the similar problem as in Section 10.6.2.

Brief Description of the Naïve Bayes Classifier:

1. Load the data from CSV files, merge them and store them in a dataset.
2. Optional Step: Re-index the dataframe if you are interested in partial dataset so that the data obtained becomes randomized using following syntax: `df = df1.reindex(np.random.permutation (df1.index))`
3. Extract `Theme_id` and `Color_id` features from dataset and store them in `X`.
4. Convert year field to binary feature. The year having ‘`num_parts`’ more than 800 as ‘1’ and year having ‘`num_parts`’ less than or equal to 800 as ‘0’.
5. Split dataset in training set and test set so that machine can be trained using `X_train` and `Y_train`.
6. `test_size` in float represents the proportion of the dataset to include in the test split, here taken as 0.33(33%). `random_state`, is an integer or None, used for initializing the internal random number generator, which decides the splitting of data into train and test sets. Default is None. If `random_state` is None, then a randomly-initialized `RandomState` object is returned. If `random_state` is an integer, then it is used to seed a new `RandomState` object.
7. Use feature scaling for `X_train`.
8. Import GaussianNB from `sklearn.naive_bayes`
9. Create a classifier and fit training set to it.

10. Make a Prediction—Estimate the accuracy of the model by making predictions for each data instance in the test dataset.
11. Estimate Accuracy—Evaluate the accuracy of the model's predicted values by comparing the two arrays (`test_labels` vs. `preds`).
12. Use the `sklearn` function `accuracy_score()` to determine the accuracy of your machine learning classifier. (optional)
13. Visualize the training and test result sets. Here we have drawn a curve and created two sections: One section for those themes and colors that are using more than 800 and other section for those themes and colors that are not having more than 800 number of parts.

Objective of the Program The Naïve Bayes classifier example presented in the section classifies input dataset on the basis of number of parts (more than 800 and less than-equal to 800) in a set. Features selected are `theme_id` and `color_id`. The program plots the resultant classes using scatter plot and contours. The classes are represented using different colors. The entire dataset is considered. User can experience the followings: handling data, summarizing data to present training and test datasets, making a prediction, evaluating accuracy and visualization of classes in the given example. Use the complete code for standalone implementation of the Naïve Bayes algorithm.

Input file(s) `sets.csv`, `inventories.csv`, `inventory_parts.csv` and `colors.csv`

Following is the complete code of complete code of Naïve Bayes classifier

```
#import the numpy, pandas, matplotlib and sklearn library
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.naive_bayes import GaussianNB

#Read csv files and extract data in dataframes df1, df2,
df3 and df4
df1 = pd.read_csv('/home/.../sets.csv')
df2 = pd.read_csv('/home/.../inventories.csv')
df3 = pd.read_csv('/home/.../inventory_parts.csv')
df4 = pd.read_csv('/home/.../colors.csv')
```

```
#Join/Merge all the dataframes into a new dataframe say, df
df1 = pd.merge(df1, df2, left_on="set_num",
right_on="set_num")
df1 = pd.merge(df1, df3, left_on="id",
right_on="inventory_id")
df1 = pd.merge(df1, df4, left_on="color_id", right_on="id")
df1 = pd.DataFrame(df1, columns=['year', 'theme_id',
'color_id', 'is_trans' , 'num_parts'])

def Build_Data_Set(features= ["theme_id", "color_id"]):
    df = df1
    X = df.iloc[:, [1, 2]].values
    df['year'] = np.where(df['num_parts']>800, '1', '0')
    y = df['year'].values
    return X, y

# Naïve Bayes Classification Code begins from here
X, y = Build_Data_Set()

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

#Define Gaussian Naïve Bayes Classifier
classifier = GaussianNB()
```

```
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the Training set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2,
              classifier.predict(np.array([X1.ravel(),
                                            X2.ravel()]).T).reshape(X1.shape),
              alpha = 0.5, cmap = ListedColormap(('grey', 'blue')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j, s = 7)
plt.title('Classifier(Training set)')
plt.xlabel('Theme Id')
plt.ylabel('Color Id')
plt.legend()
plt.show()

#Estimate Accuracy
from sklearn.metrics import accuracy_score
```

```

print("Prediction = ", y_pred)
print("Accuracy = ", accuracy_score(y_test, y_pred))

# Visualizing the Test set results
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2,
              classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
              alpha = 0.5, cmap = ListedColormap((‘cyan’, ‘yellow’)))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap((‘red’, ‘green’))(i), label = j, s = 7)
plt.title(‘Naive Bayes(Test set)’)
plt.xlabel(‘Theme Id’)
plt.ylabel(‘Color Id’)
plt.legend()
plt.show()

```

Output of Naïve Bayes classifier

Figure 10.8 shows output of Naïve Bayes classifier. The rows of df dataframe are shuffled and selected first 50,000 rows only ($df = df[1:50000]$). The Green data points are themes and colors with more than 800 number of parts and Red for less than equal to 800 parts.

10.6.4 Regression Analysis Algorithms

The following subsections consider data fitting using Linear Regression and Polynomial Regression Functions:^{7,8,9}

10.6.4.1 Fitting a Linear Regression Function

Linear regression model predicts a direct proportional (linear) relationship between the dependent variable (plotted on the vertical or y-axis) and the predictor variables (plotted on the x-axis). Figure 10.9 shows linear regression example. The figure shows that the model produces a straight line.

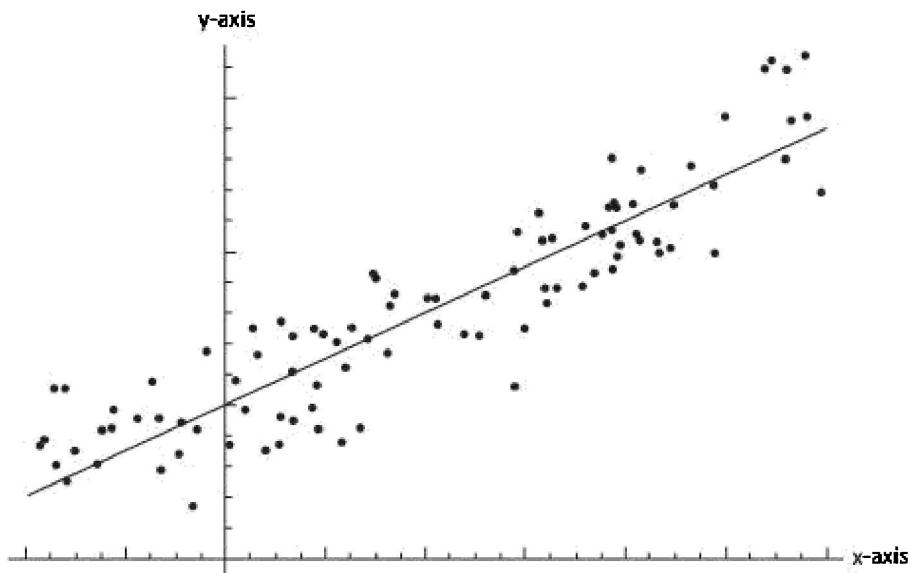


Figure 10.9 Example of linear regression

Selection of Predictor Variable The Lego toys dataset contains the Lego Parts/Sets/Colors and Inventories of each official Lego set. The dependent variable we are using is the count of colors (`color_id` aggregate), that provides the augmentation of colors with respect to time or not.

Brief Description of the Linear Regression Algorithm Code

1. Perform basic imports first. Import Pandas for reading the data from csv files and creating pandas dataframes for doing data analysis, then matplotlib for graphing, and `linear_model` from `sklearn`.
2. Next, read the input csv files into individual dataframes and merge them to get a final dataframe `df` of year and `color_id` attributes.
3. Filter the undesired data, if required. Here year 2017 is not being included since it has data up to July only. The incomplete data (not the full year data) may result into wrong prediction.
4. Calculate the color usage in each year from 1950 to 2016 and store the year and

count value in a new dataframe called yearwise_colour_count.

5. Next, extract the independent variable or predictor (X) and dependent variable or target (Y) values from data. For example, here the color usage depends upon the year value. Thus, year is independent variable while color count is dependent.
6. Apply the linear regression function from linear_model library to obtain regression line with slope, intercept values and Next, use fit () function to fit the predictor and target variables and to obtain a model.
7. The model can be used to predict new values. Predict the color count expected in future year value (say, 2020 as shown in the example).
8. Visualize the scatter graph and regression line using plot function as given in the example.

There are multiple ways to perform linear regression in Python , use the:

1. Scipy
2. Statsmodels
3. Scikit-learn library.

The following example uses Scikit-learn library for linear regression.

Objective of the Program The linear regression example presented in the section demonstrates year wise growth of color usage. Features selected are year and color_id. The program plots the aggregated counts of year field using scatter plot. The line of regression will be shown to be exploited for predicting future count value. The predictor variable will be the year value. Since the database contains Lego sets from 1950 to July 2017, we aim to predict the increase/decrease of the colors in future (say, Year 2020)

Input file(s) sets.csv, inventories.csv, inventory_parts.csv and colors.csv

Following is the code for linear regression:

```
#import pandas, matplotlib and sklearn libraries
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import linear_model
```

```
#Read csv file and extract data in dataframes df1, df2, df3  
and df4  
  
df1 = pd.read_csv("/home/.../sets.csv")  
df2 = pd.read_csv("/home/.../inventories.csv")  
df3 = pd.read_csv("/home/.../inventory_parts.csv")  
df4 = pd.read_csv("/home/.../colors.csv")  
  
#Join/Merge all the dataframes into a new dataframe say,  
df1##  
  
df1 = pd.merge(df1, df2, left_on="set_num", right_on =  
"set_num")  
df1 = pd.merge(df1, df3, left_on="id", right_on =  
"inventory_id")  
df1 = pd.merge(df1, df4, left_on="color_id", right_on =  
"id")  
  
#Extract the year and color_id features from merged  
dataframe df1  
  
#and Store in dataframe df  
df = pd.DataFrame(df1, columns=['year','color_id'])  
#Year 2017 may be ignored since it has data up to July  
#(which is not the full year data)  
df = df[(df['year'] <2017)]  
# Count the colour usage in each year from & store as  
# in yearwise_colour_count  
yearwise_colour_count = pd.DataFrame({'count' :  
df.groupby('year')[ "year"].count() }).reset_index()  
#Extract X(predictor) and Y(target)values from  
yearwise_colour_count  
X = yearwise_colour_count.iloc[:, :-1].values  
y = yearwise_colour_count.iloc[:,1].values  
  
#Obtain Line of regression with slope, intercept and other  
values  
regression =linear_model.LinearRegression()
```

```

# feed the linear regression with the train data to obtain
# a model.

regression.fit(X,y)
#Calculate slope and intercept
slope= regression.coef_[0]
intercept= regression.intercept_
#Display slope and intercept
print("Slope = ", round(slope,2), ", Intercept = ",
round(intercept,2))

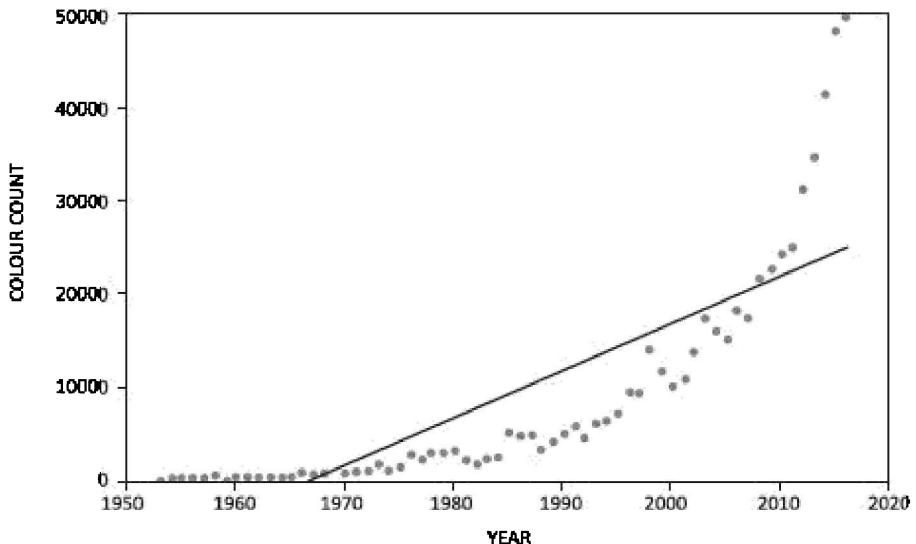
# Obtain prediction for the year 2020
newX = 2020
newy = newX*slope + intercept #Using Line Equation
#Y=X*SLOPE+INTERCEPT

#output of Colour Count Predicted in year 2020
print("Colour Count predicted in ", newX, " = ",
round(newy,0))

# plot the scatter graph and regression line
plt.plot(X, y, '.', color='gray')
plt.xlabel("YEAR")
plt.ylabel("COLOUR COUNT")
#Display Year on X axis that varies from Year 1950 to 2016
#Display Colour Count on Y axis that varies from 0 to
50,000
plt.axis([1950,2020,0,50000])
#Display line of Regression
plt.plot(X, X*slope+intercept, 'k')
plt.show()

```

Output of the Linear Regression Figure 10.10 shows linear regression output



Slope = 508.66 , Intercept = -1000662.09

Colour Count predicted in 2020 = 26835

Figure 10.10 Linear regression output

Function for regression line in Scikit-learn library used in above example is as follows:

```
from sklearn import linear_model
regression =linear_model.LinearRegression()
regression.fit(X, y)
slope= regression.coef_[0]
intercept= regression.intercept_
```

Scipy Library also provides a function for linear regression analysis. This library provides the `linregress` function. Replace the above written statements with the following statements in the complete code of linear regression while you are using `scipy` library:

```
from scipy import stats
slope, intercept, r_values, p_values, std_err =
stats.linregress(X, y)
```

The output will be the same as Figure 10.10.

Salient Observation The output illustrated in Figure 10.10 is inappropriate. When the color count value is 49,904 in 2016, the predicted value of 2020 being 26,834.690 is misleading. We can observe that the data points are following a curve and the regression line in the figure is not fitting the curve.

The solution to this is a polynomial regression (also termed as quadratic regression) that has quality to fit a non-linear model to the data. Curve fitting is the process of constructing a curve, or mathematical function that has the best fit to a series of data points.

Linear regression equation is:

Dependent variable = Constant + Parameter * Independent variable1 + ... + Parameter * Independent variable n

$$y = b_0 + b_1 x_1 + \dots + b_n x_n \quad (10.1)$$

The regression equation (10.1) is linear in the parameters. However, it is possible to model curvature using this equation. The function parameters are linear but one can raise independent variable by an exponent to fit a curve. For example, by squaring the independent variable (Equation 10.2), the model can follow a U-shaped curve.

$$y = b_0 + b_1 x_1 + \dots + b_n x_n^2 \quad (10.2)$$

While the independent variable is squared, the model is still linear in the parameters. Linear models can also contain log terms and inverse terms to follow different kinds of curves and yet continue to be linear in the parameters.

10.6.4.2 Fitting a Polynomial Regression Function

Brief description of the polynomial regression algorithm code is as follows:

1. Perform basic imports first. Import Pandas for reading the data from csv files and creating pandas dataframes for doing data analysis, then matplotlib for visualization through graphs.
2. Next, read the input csv files into individual dataframes and merge them to get a final dataframe df of year and color_id attributes.
3. Filter the undesired data, if required. Here, year 2017 is not being included since it has data up to July only. The incomplete data (not the full year data) may result into wrong prediction.
4. Calculate year wise number of colors usage and store it in a new dataframe called data here.
5. Next, extract the independent variable or predictor (X) and dependent variable or target (Y) values from data. For example, here the color usage depends upon the year value. Thus, year is independent variable while color count is dependent.

6. Define the degree of polynomial, say 2, 3, 4 and 5
7. Apply the polyfit function from numpy library with defined degree of polynomial. The function outputs the regression coefficients.^{10,11}
8. Use the coefficients to obtain a polynomial. The polynomial models the regression curve that fits the independent variables.
9. The model can be used to predict new values. Predict the color count expected in future year value (say, 2020 as shown in the example)
10. Visualize the scatter graph and regression line using plot function as given in this example.

Objective of the Program The polynomial regression example presented in the section demonstrates year-wise growth of color usage. Features selected are year and color_id. The program plots the aggregated counts of year field using scatter plot. The growth actually forms a curve when drawn against year. The regression polynomial fits the curve and predicts the future count value (Proven to be more accurate than linear regression for this particular example).

Input file(s) sets.csv, inventories.csv, inventory_parts.csv and colors.csv

The complete source code for polynomial Regression using Scipy Library is as follows:

```
#import numpy, pandas, matplotlib and sklearn libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math # For trunc() to truncate the decimal values

#Read csv file and extract data in dataframes df1, df2, df3
and df4
df1 = pd.read_csv("/home/.../sets.csv")
df2 = pd.read_csv("/home/.../inventories.csv")
df3 = pd.read_csv("/home/.../inventory_parts.csv")
df4 = pd.read_csv("/home/.../colors.csv")

#Join/Merge all the dataframes into a new dataframe say,
df1#
```

```

df1 = pd.merge(df1, df2, left_on="set_num", right_on =
"set_num")
df1 = pd.merge(df1, df3, left_on="id", right_on =
"inventory_id")
df1 = pd.merge(df1, df4, left_on="color_id", right_on =
"id")
df = pd.DataFrame(df1, columns=['year','color_id'])
#Year 2017 may be ignored since it has data up to July
#: Not the full year data
df = df[(df['year'] <2017)]

# Count the colour usage in each year and store as
yearwise_colour_count
yearwise_colour_count = pd.DataFrame({'count' :
df.groupby('year')[ "year"].count()}).reset_index()

#Extract X(predictor) and Y(target)values from
yearwise_colour_count
X = yearwise_colour_count ['year'].values
y = yearwise_colour_count ['count'].values
#Obtain minimum & maximum of X and y values for drawing
axes
min_x= X.min()
max_x= X.max()
min_y= y.min()
max_y= y.max()

#Initialize degree of polynomial, say 2,3,4,5
degree_of_polynomial = 2
#Fit the polynomial regression and Obtain the coefficients
values
coefs = np.polyfit(X, y, degree_of_polynomial)
#Create a polynomial from the obtained coefficients
p = np.poly1d(coefs)

```

```

# prediction # Obtain prediction for the year 2020
newx = 2020
newy = p(newx)

#output of Colour Count Predicted in year 2020
print("Colour Count predicted in ",newX, "= ", 
math.trunc(round(newy,0)))

# plot the scatter graph and regression curve
plt.plot(X, y, "bo", markersize= 2)
plt.plot(X, p(X), "r-", color='k') #p(X) evaluates the
polynomial at X
plt.plot(X, y, '.', color='gray')
plt.xlabel("YEAR")
plt.ylabel("COLOUR COUNT")
#Display Year on X axis that varies from min_x(1950) to
max_x(2016)
#Display Colour Count on Y axis that varies from min_y(0)
to max_y
plt.axis([min_x,max_x,min_y,max_y])
plt.show()

```

Output of the Polynomial Regression with Different Degrees of Polynomial

Figure 10.11 shows polynomial regression outputs for 2 and 3 degree polynomials. Figure 10.12 shows polynomial regression outputs for 4 and 5 degrees. Figure 10.13 shows polynomial regression output for 6 degree.

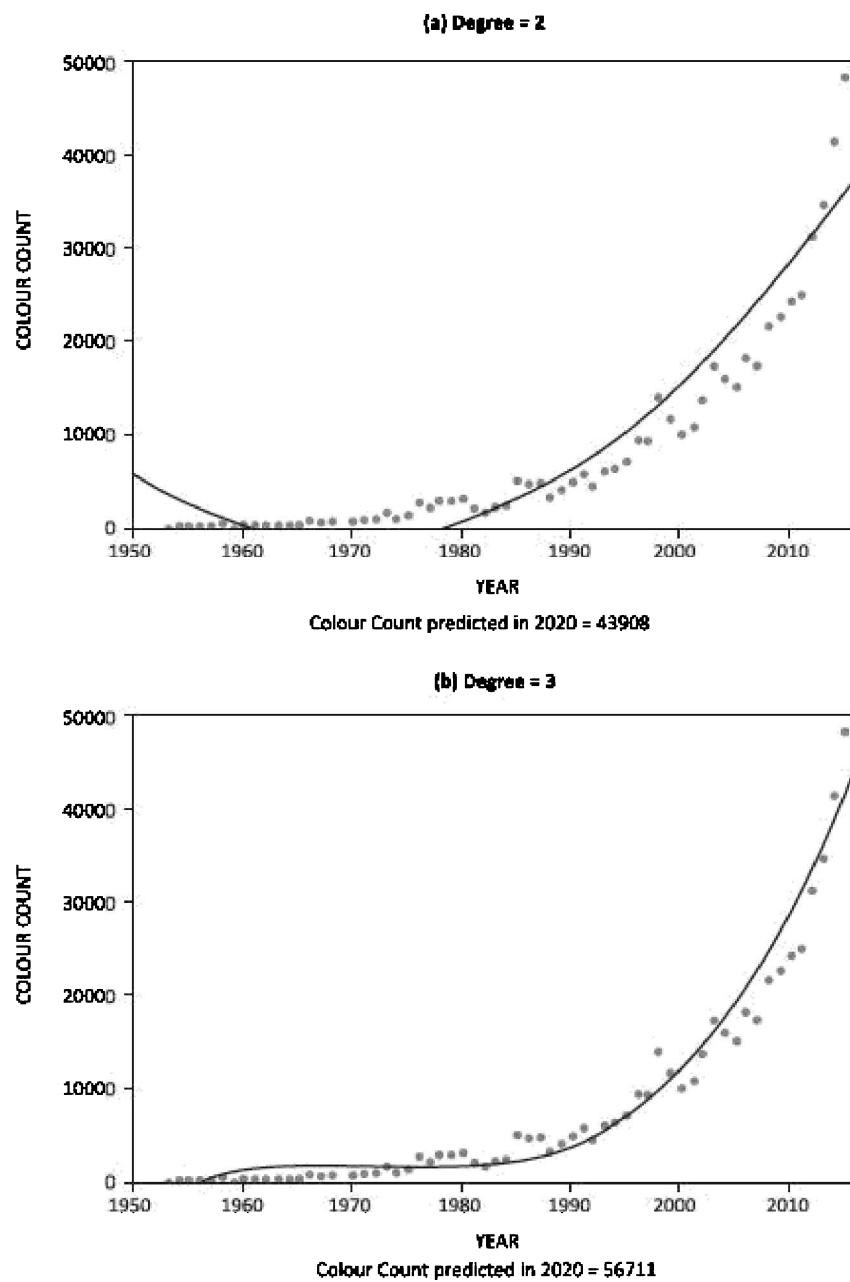
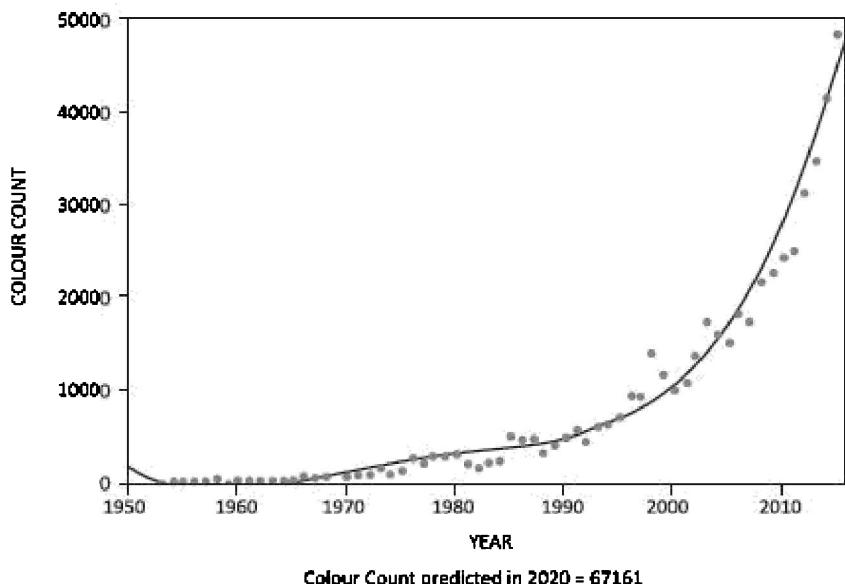


Figure 10.11 Polynomial regression outputs for 2 and 3 degrees

(c) Degree = 4



(d) Degree = 5

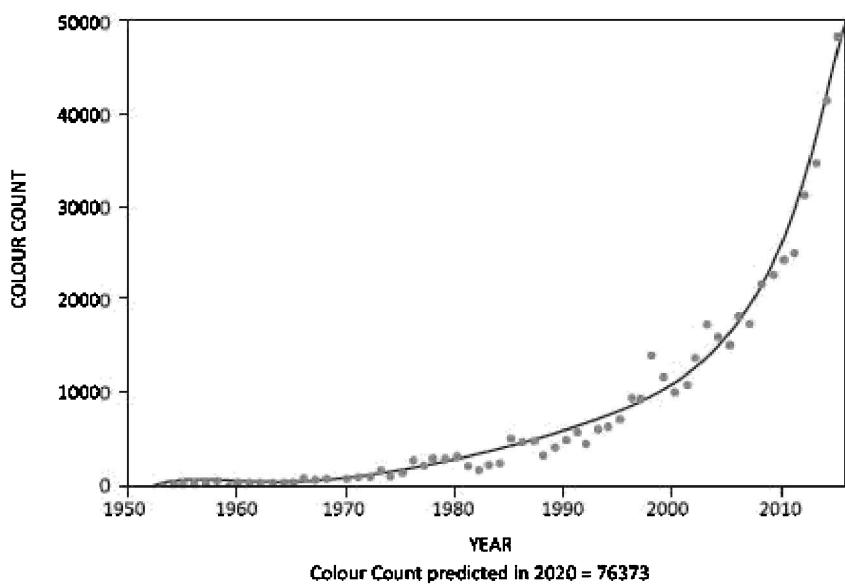


Figure 10.12 Polynomial regression outputs for 4 and 5 degrees

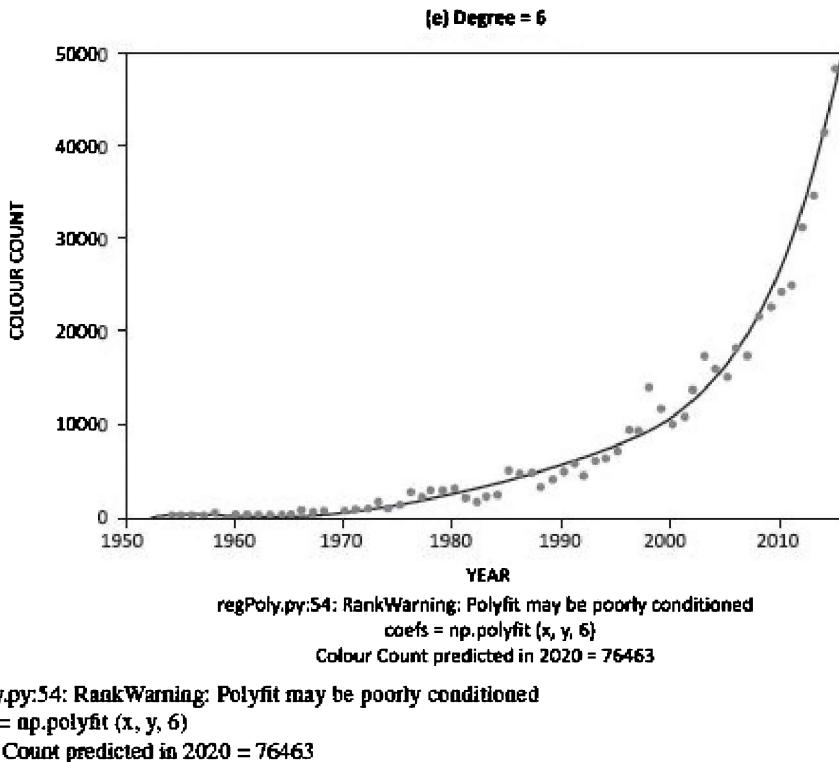


Figure 10.13 Polynomial regression output for 6 degree

Practice Exercises

10.1 Explore the dataset of the New Car Sales in Norway. The dataset is open source and downloadable from at www.kaggle.com¹² The dataset contains monthly car sales for 2007–2017 by make and most popular models.

Dataset includes three csv files:

1. **Norway_new_car_sales_by_make.csv:**

Monthly sales of new passenger cars by make (manufacturer brand)

1. Year – year of sales
2. Month – month of sales
3. Make – car make (e.g. Volkswagen, Toyota, Tesla)
4. Quantity – number of units sold
5. Pct – percent share in monthly total

2. Norway_new_car_sales_by_model.csv:

Monthly summary of top-20 most popular models (by make and model)

1. Year – year of sales
2. Month – month of sales
3. Make – car make (e.g. Volkswagen, Toyota, Tesla)
4. Model – car model (e.g. BMW-i3, Volkswagen Golf, Tesla S75)
5. Quantity – number of units sold
6. Pct – percent share in monthly total

3. Norway_new_car_sales_by_month.csv:

Summary statistics for car sales in Norway by month

1. Year – year of sales
2. Month – month of sales
3. Quantity – total number of units sold
4. Quantity_YoY – change YoY in units
5. Import – total number of units imported (used cars)
6. Import_YoY – change YoY in units
7. Used – total number of units owner changes inside the country (data available from 2012)
8. Used_YoY – change YoY in units
9. Avg_CO2 – average CO2 emission of all cars sold in a given month (in g/km)
10. Bensin_CO2 – average CO2 emission of bensin-fueled cars sold in a given month (in g/km)
11. Diesel_CO2 – average CO2 emission of diesel-fueled cars sold in a given month (in g/km)
12. Quantity_Diesel – number of diesel-fueled cars sold in the country in a given month
13. Diesel_Share – share of diesel cars in total sales (Quantity_Diesel / Quantity)
14. Diesel_Share_LY – share of diesel cars in total sales a year ago
15. Quantity_Hybrid – number of new hybrid cars sold in the country (both PHEV

and BV)

16. Quantity_Electric – number of new electric cars sold in the country (zero emission vehicles)

17. Import_Electric – number of used electric cars imported to the country (zero emission vehicles)

Norway new car sales dataset can be used for analyzing and predicting future car sales. Explore the dataset to solve the following problems for analysis, prediction and visualization using Sklearn and PySpark:

1. Print year-wise total car sales and visualize the output (Hint: use bar chart for Year vs. total car sales).
2. Print monthly total car sales and visualize for a specified year.
3. Print monthly total car sales from 2007 to 2017 and visualize them to represent the month numbers (1 for Jan 2 for Feb) and total car sales value. (Hint: Use bar chart). Also find the month for number of highest and lowest car sales.
4. Calculate the total amount of the sales for each manufacturer from 2007 to 2017. Find the top 10 manufacturers based on the total sale and visualize the output. (Hint: Sort make-wise total car sales and visualize them using bar chart).
5. Draw pie chart for the sales of all the models of “Toyota” in year 2012.
6. Find which model of each manufacturer has the highest sales in year 2015.
7. Find which model of each manufacturer has the highest sales during 2007 to 2017.
8. Find which model of each manufacturer has the lowest sales during 2007 to 2017.
9. Compare car models with percentage shares.
10. Predict (forecast) the car sales for all the months of 2020 using the month-wise car sales quantity from the Jan 2007 to Jan 2017.
11. Plot the sales of new cars and sales of the diesel cars to see the comparison. Similarly, plot the sales of new car and electric cars. (Hint: Use line chart).
12. Calculate year-wise share of diesel car sales in total sales.
13. Compare year-wise average consumption of CO₂ emission of all cars sold with year-wise average consumption of CO₂ emission in benzene-fueled cars sold and diesel-fueled cars sold.
14. Calculate and visualize year-wise new and used (import) car sales to compare the

statistics.

15. Calculate and visualize year-wise sales of all used (import) car and sales of electric-used cars (import_electric) to make a comparison.
 16. Predict the rise of green vehicles in 2018. (Hint: Green Vehicle = Import_Electric + Quantity_Hybrid + Quantity_Electric).
 17. Forecast and visualize the diesel market share. State whether it represents growth or reduction in the sales.
 18. Rank top 10 car brands. Visualize the year-wise result using line graph.
-

1 <https://hadoop.apache.org/> Hadoop

2 <https://www.edureka.co/blog/hadoop-3/>

3 <https://rebrickable.com/about/>

4 LDB— The LEGO Parts/Sets/Colors and inventories of every official LEGO set, available at: <https://www.kaggle.com/rtatman/lego-database>

5 <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

6 <https://sadanand-singh.github.io/posts/svmpython/>

7 <http://www.learndatasci.com/predicting-housing-prices-linear-regression-using-python-pandas-statsmodels/>

8 <https://codefying.com/2016/08/18/two-ways-to-perform-linear-regression-in-python-with-numpy-and-sk-learn/>

9 <http://statisticsbyjim.com/regression/difference-between-linear-nonlinear-regression-models/>

10 <https://autarkaw.org/2008/07/05/finding-the-optimum-polynomial-order-to-use-for-regression/>

11 <https://in.mathworks.com/help/matlab/ref/polyfit.html?requestedDomain=true>

12 <https://www.kaggle.com/dmi3kno/newcarsalesNorway>

Bibliography

A. PRINTED AND E-BOOKS

- [1] Anand Rajaraman and Jeffrey David Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2012
- [2] Andrew Gelman, Jennifer Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press, 2007
- [3] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, Donald B. Rubin, *Bayesian Data Analysis*, 3rd Ed., CRC Press, 2013
- [4] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn and Tensor Flow: Concepts, Tools, and Techniques to Build Intelligent System*, O'Reilly Media, 2017
- [5] Bernard Marr, *Big Data in Practice: How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary Results*, Wiley, UK, 2016
- [6] Carl Shan, William Chen, Henry Wang, Max Song, *The Data Science Handbook: Advice and Insights from 25 Amazing Data Scientists*, Paperback, The Data Science Bookshelf, 2015
- [7] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006
- [8] David Dietrich, Barry Heller, Beibei Yang, *Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*, EMC Education Services, Wiley, Indianapolis, 2015
- [9] David A. Freedman, *Statistical Models: Theory and Practice*, Paperback, 2nd Ed., Cambridge University Press, 2009
- [10] David J.C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003
- [11] David Loshin, *Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph*, Morgan Kaufmann (Elsevier), MA 02451, 2013
- [12] E. T. Jaynes, *Probability Theory, The Logic of Science*, Cambridge University Press, 2003
- [13] Edward Capriolo, *Cassandra High Performance Cookbook*, Packt, 2011
- [14] Edward R. Tufte, *Visual Explanations: Images and Quantities, Evidence and Narrative*, Google Books, 1997
- [15] Erik Brynjolfsson, Andrew McAfee, *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*, W. W. Norton, 2014 (Also available at ACM Digital Library)
- [16] Erwin Kreyszig, *Advanced Engineering Mathematics*, 12th Ed., Wiley, 2016 (Paperback Edition, Wiley, 2014)
- [17] Francois Chollet, *Deep Learning with Python*, Manning, 2018
- [18] Garry Kasparov, Mig Greengard, *Deep Thinking— Where Artificial Intelligence Ends and Human Creativity Begins*, Google Books, 2017
- [19] Garry Turkington, *Hadoop: Beginner's Guide*, Packt, 2013
- [20] George Casella, Roger L. Berger, *Statistical Inference*, 2nd Ed., Cengage Learning, 2006
- [21] Giancarlo Zaccone, Md. Rezaul Karim, *Deep Learning with Tensor Flow*, 2nd Ed., Packt, 2018
- [22] Hadley Wickham, *ggplot2: Elegant Graphics for Data Analysis*, 2nd Ed., Springer, 2016

- [23] I. Craig, Allen Thornton, *Mathematical Statistics*, 5th Ed., Macmillan, NY, 1989 [1st E. 1958]
- [24] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning* (eBook), MIT Press, 2017
- [25] Ian Robinson, *Graph Databases*, Paperback, O'Reilly Media, 2nd Ed., 2015
- [26] Jeff Hawkins, Sandra Blakeslee, *On Intelligence, How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines*, Paperback, St. Martin Griffin, 2005 [Prentice Hall, New Jersey, 2004]
- [27] John A. Rice, *Mathematical Statistics and Data Analysis*, 3rd Ed., Thomson Learning, Is Indian Reprint, 2007
- [28] John H. Mathews, Kurtis D. Fink, *Numerical Methods Using Matlab*, Paperback, Pearson, 2015
- [29] John W. Foreman, *Data Smart: Using Data Science to Transform Information into Insight*, Paperback, Wiley, Indianapolis, 2014
- [30] Joshan D. Suereth, *Scala in Depth*, Manning, 2012
- [31] Kevin P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, 2012
- [32] M. Gopal, *Applied Machine Learning*, McGraw Hill, India, 2018
- [33] Malcom Gladwell, *Outliers: The Story of Success*, Penguin, London, 2013
- [34] Max Kuhn, Kjell Johnson, *Applied Productive Modeling*, Springer, 2013
- [35] Maxim Lapan, *Deep Reinforcement Learning Hands-On*, Packt, 2018
- [36] Michael Frampton, *Complete Guide to Open Source Big Data Stack*, Apress, 2018
- [37] Michael G. Milton, *Head First Data Analysis: A Learner's Guide to Big Numbers, Statistics, and Good Decisions*, O'Reilly Media, 2010
- [38] Mike Barlow, *Realtime Big Data Analytics: Emerging Architecture*, O'Reilly Media, 2014
- [39] Mike Driscoll, *Python Interviews*, Packt, 2018
- [40] Mike Frampton, *Mastering Apache Spark*, Packt, 2017
- [41] Mohammed Guller, *Big Data Analytics with Spark: A Practitioner's Guide to Using Spark for Large Scale Data Analysis*, Apress, 2015
- [42] Nathan Marz, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, Paperback, Manning, 2015
- [43] Niranjan Raychaudhuri, *Scala in Action*, Manning, 2012
- [44] Paul C. Zikopoulos, Chris Eaton, Dirk deRoos, Tom Deutsch, George Lapis, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, McGraw Hill, NY 2012 (India ed., 2015)
- [45] Peter Norvig, *Paradigms of Artificial Intelligence Programming, Case Studies in Common LISP*, Paperback, Morgan Kaufmann, California, 1992
- [46] Philip E. Tetlock, *Superforecasting: The Art and Science of Prediction*, Broadways Books, N.Y. (Penguin), 2015
- [47] Prateek Joshi, *Python Machine Learning Cookbook*, Packt, 2016
- [48] Richard W. Hamming, *Numerical Methods for Scientists and Engineers*, McGraw-Hill, New York, 1973 (2nd Ed. 1987, Dover. Available at the ACM Digital Library)
- [49] Robert J. Kabacoff, *R in Action— Data Analysis and Graphics with R*, Manning, 2011
- [50] Robert V. Hogg, I. Craig, Allen T. Craig, *Introduction to Mathematical Statistics*, 4th Ed., McMillan N.Y., 1978
- [51] Romeo Kienzler, *Mastering Apache Spark 2.x*, 2nd Ed., Packt, 2017

- [52] Satnam Alag, *Collective Intelligence in Action*, Manning, 2008
- [53] Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman, *Mahout in Action*, Manning, 2012
- [54] Sean T. Allen, Matthew Jankowski, Peter Pathirana, *Storm Applied*, Manning, 2015 (Also available at ACM Digital Library)
- [55] Sebastian Raschka, Vahod Mirjalili, *Python Machine Learning*, 2nd Ed., Packt, 2017
- [56] Seema Acharya, Subhasini Chellappan, *Big Data and Analytics*, Wiley, 2015
- [57] Seema Acharya, *Data Analysis using R*, McGraw Hill, India, 2018
- [58] Steven C. Chapra, Raymond Canale, *Numerical Methods for Engineers*, Google Books, 1985
- [59] Stuart J. Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, Springer, New York, 1995
- [60] Tanmay Deshpandey, *Hadoop Real World Solutions*, 2nd Ed., Packt, 2016
- [61] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd Ed., Springer, 2001
- [62] Tomasz Draba, Denny Lee, *Learning PySpark*, Packt, Manning, 2017
- [63] Trevor Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2001
- [64] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*, 3rd Ed., Cambridge Press, 2002 (South Asian Ed. Foundation Books, India, 2002; Originally Published in FORTRAN, Available as pdf open source)
- [65] Yifeng Jiang, *HBase Administration Cookbook*, Packt, 2012

B. WEBSITE REFERENCES

- [1] <https://aws.amazon.com/machine-learning/amis/> AWS Deep Learning AMIs, *Pre-configured Environments to Quickly Build Deep Learning Applications*
- [2] <https://arxiv.org/abs/1709.02840> (Cornell University Library), Osvaldo Simeone, *A Brief Introduction to Machine Learning for Engineers*, 2018
- [3] <https://web.stanford.edu/~hastie/Papers/ESLII.pdf>, Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd Ed., 2001
- [4] https://vuquangnguyen2016.files.wordpress.com/2018/03/applied-predictive-modeling-max-kuhn-kjell-johnson_1518.pdf, Max Kuhn, Kjell Johnson, *Applied Productive Modeling*, Springer, 2013
- [5] <http://www.freetechbooks.com/the-elements-of-data-analytic-style-t1257.html>, Jeff Leek, *The Elements of Data Analytic Style*, 2016
- [6] <https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf>, Stuart J. Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, 2005
- [7] <https://www.thedatasciencehandbook.com>, Carl Shan, William Chen, Henry Wang, Max Song, *The Data Science Handbook: Advice and Insights from 25 Amazing Data Scientists*, Paperback, 2015
- [8] https://tecnoclub4u.files.wordpress.com/2015/05/hadoop_-beginners-guide.pdf, *Hadoop Beginners Guide*, Packt
- [9] <http://barbie.uta.edu/~jli/Resources/MapReduce&Hadoop/Hadoop%20MapReduce%20Cookbook.pdf>, Srinath Perera Thillina Gunarathne, *Hadoop MapReduce CookBook*, Packt

- [10] <http://www.ieee.org.ar/downloads/Srivastava-tut-pres.pdf>, Jaideep Srivastava, University of Minnesota USA, *Web Mining: Accomplishments & Future Directions*:
- [11] <https://www.isical.ac.in/~acmsc/WBDA2015/slides/hg/Oreilly.Hadoop.The.Definitive.Guide.3rd.Edition.Jan.2012.pdf>
- [12] [https://www.isical.ac.in/~acmsc/WBDA2015/slides/hg/Oreilly, Tom White, *Hadoop: The.Definitive.Guide.* 3rd. Edition, Jan. 2012 pdf](https://www.isical.ac.in/~acmsc/WBDA2015/slides/hg/Oreilly, Tom White, Hadoop: The.Definitive.Guide. 3rd. Edition, Jan. 2012 pdf)
- [13] https://websites.pmc.ucsc.edu/~fnimmo/eart290c_17/NumericalRecipesinF77.pdf, William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing*, Cambridge Press, 2nd Ed., Reprinted with Corrections/New Software version 2.08, 1997

C. CHAPTER-WISE WEBSITE REFERENCES

Chapter 1

- ¹ <http://www.gartner.com/it-glossary/big-data>
- ² <https://statswiki.unece.org/display/bigdata/Classification+of+Types+of+Big+Data>
- ³ <https://www.ibm.com/developerworks/library/bd-archpatterns1/>
- ⁴ <https://docs.microsoft.com/en-us/sql/odbc/reference/data-sources>
- ⁵ https://docs.oracle.com/cd/E17984_01/doc.898/e14695/undrstnd_datasources.htm
- ⁶ https://www.ibm.com/support/knowledgecenter/en/SSMPHH_9.5.0/com.ibm.guardium95.doc/common_tools/topics/datasources.html
- ⁷ <http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-copyactivity.html>
- ⁸ <http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-snsalarm.html>
- ⁹ <https://support.rackspace.com/how-to/cloud-database-instance-parameters/>
- ¹⁰ <https://cloud.google.com/bigquery/docs/loading-data>

Chapter 2

NIL

Chapter 3

- ¹ <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC#LanguageManualORC-ORCFfileFormat>
- ² <http://www.semantikoz.com/blog/orc-intelligent-big-data-file-format-hadoop-hive/>

Chapter 4

NIL

Chapter 5

- ¹ <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC#LanguageManualORC-ORCFfileFormat>
- ² <https://www.tecmint.com/install-java-jdk-jre-in-linux/>
- ³ <https://www.vultr.com/docs/how-to-manually-install-java-8-on-ubuntu-16-04>

⁵ <https://data-flair.training/blogs/create-rdds-in-apache-spark/>

⁶ <http://data-flair.training/forums/topic/how-to-create-rdd>

Chapter 6

¹ [https://en.wikipedia.org/wiki/Kernel_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics))

³ https://en.wikipedia.org/wiki/Levenshtein_distance

Chapter 7

⁴ <http://algo.inria.fr/flajolet/Publications/FlMa85.pdf>

Chapter 8

¹ <https://neo4j.com/product/>

² <http://www.systemg.research.ibm.com/>

Chapter 9

¹ http://www.nactem.ac.uk/brochure/NaCTeM_Brochure.pdf

² https://www.ibm.com/support/knowledgecenter/en/SS3RA7_18.1.1/ta_guide_ddita/textmining/shared_entities/tm_intro_tm_defined.html

³ <https://blogs.aws.amazon.com/bigdata/post/Tx22THFQ9MI86F9/Applying-Machine-Learning-to-Text-Mining-with-Amazon-S3-and-RapidMiner>

⁴ https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

⁵ <http://papers.cumincad.org/data/works/att/2873.content.pdf> “*The Anatomy of a Large-Scale Hypertextual Web Search Engine*” Sergey Brin Lawrence Page, 1998

Chapter 10

¹ <https://hadoop.apache.org/> Hadoop

² <https://www.edureka.co/blog/hadoop-3/>

³ <https://rebrickable.com/about/>

⁴ LDB— The LEGO Parts/Sets/Colors and inventories of every official LEGO set, available at: <https://www.kaggle.com/rstatman/lego-database>

⁵ <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>

⁶ <https://sadanand-singh.github.io/posts/svmpython/>

⁷ <http://www.learndatasci.com/predicting-housing-prices-linear-regression-using-python-pandas-statsmodels/>

⁸ <https://codefying.com/2016/08/18/two-ways-to-perform-linear-regression-in-python-with-numpy-and-sklearn/>

⁹ <http://statisticsbyjim.com/regression/difference-between-linear-nonlinear-regression-models/>

¹⁰ <https://autarkaw.org/2008/07/05/finding-the-optimum-polynomial-order-to-use-for-regression/>

¹¹ <https://in.mathworks.com/help/matlab/ref/polyfit.html?requestedDomain=true>

¹² <https://www.kaggle.com/dmi3kno/newcarsalesNorway>

D. PRINTED/E-JOURNAL PAPER REFERENCES

- [1] IEEE Transactions on Big Data, Quarterly, Since 2015
- [2] Elsevier Big Data Research Journal, Latest Issue Volume 13, Sept 2018, Quarterly Since 2016, (First Issue 2011)
- [3] Springer, Journal of Big Data, Open Access, Since 2014
- [4] BMC Series Springer Nature, Big Data Analytics

Answers to Multiple Choice Questions

CHAPTER 1

1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
(d)	(e)	(b)	(a)	(c)	(d)	(b)	(d)	(a)	(b)
11.	12.	13.	14.	15.	16.				
(c)	(d)	(e)	(c)	(d)	(a)				

CHAPTER 2

1	2	3	4	5	6	7	8	9	10
(d)	(b)	(e)	(a)	(c)	(b)	(a)	(e)	(b)	(a)
11	12								
(d)	(d)								

CHAPTER 3

1	2	3	4	5	6	7	8	9	10
(a)	(a)	(e)	(b)	(d)	(b)	(a)	(d)	(b)	(b)
11	12	13	14	15.					
(a)	(e)	(d)	(e)	(b)					

CHAPTER 4

1	2	3	4	5	6	7	8	9	10
(e)	(a)	(d)	(b)	(d)	(d)	(b)	(e)	(a)	(b)
11	12								
(c)	(a)								

CHAPTER 5

1	2	3	4	5	6	7	8	9	10
(b)	(a)	(d)	(c)	(e)	(a)	(a)	(d)	(b)	(b)
11	12	13	14						
(d)	(e)	(b)	(a)						

CHAPTER 6

1	2	3	4	5	6	7	8	9	10
(a)	(e)	(d)	(b)	(d)	(a)	(d)	(e)	(e)	(a)
11	12	13	14						
(b)	(d)	(b)	(e)						

CHAPTER 7

1	2	3	4	5	6	7	8	9	10
(b)	(b)	(a)	(d)	(e)	(c)	(a)	(b)	(e)	(d)
11	12								
(a)	(b)								

CHAPTER 8

1	2	3	4	5	6	7	8	9	10
(d)	(e)	(a)	(a)	(b)	(c)	(d)	(b)	(c)	(a)

CHAPTER 9

1	2	3	4	5	6	7	8
(d)	(c)	(b)	(a)	(b)	(b)	(b)	(b)

Index

3Vs 2, 9, 10, 11

4Vs 9, 10

A

accumulator 233, 234

ACID properties 8, 35, 88, 90, 92, 93, 94, 97, 100, 115, 118, 146

AdaBoost 303, 317, 329

adapter 62, 338, 339

adhoc query 207, 213

aggregation 5, 33, 67, 73, 109, 110, 124, 126, 138

aggregation function 147

Amazon S3 *see* S3

Ambari 54, 73, 75, 76, 80, 145, 204

analytics processing framework 240

anomaly 257, 390, 392

anomaly detection 212, 240, 393, 394, 403, 442

ANOVA 262

Apache GraphX 40, 211, 397-401, 403, 404, 436, 440, 446

Apache Mahout 20, 40, 61, 65, 73, 78, 204, 251, 252, 285, 323-327, 330, 454

Apache Spark 38, 60-62, 80, 110, 115, 145, 205,
207-235, 240, 245-247

Apache Spark streaming 364

API, definition 2

application integration 6, 34, 54
application master 71
application, definition 2
Apriori algorithm 286, 287, 329, 359-362
artificial intelligence 10, 19, 252
association 262, 263, 288-291, 305, 321, 329, 378, 380-390, 392, 398, 409, 428, 438
association rule 234, 251, 285, 286, 322, 329, 358, 454
association-rule mining 280, 285, 430
authority 432-435, 438-441, 448
auto-sharding 125
availability 16, 18, 20, 35, 43, 56-58, 66, 74, 90,
91, 93, 108, 110, 117, 119, 124, 125, 130, 147, 300, 348
AVRO 29, 61, 62, 68, 135

B

bag 189, 193
bag of words 280, 413, 414, 427
BASE 87, 93, 94, 114, 117, 118, 147
batch processing 5, 13, 20, 60, 62, 78, 338, 346,
347, 364
Bayes theorem 315
Bayesian classifier 315, 326
Bayesian network 387-389, 395, 397, 403
BDAS *see* Berkeley Data Analytics Stack
belief network 253, 388, 391, 395, 396, 410, 441, 442
belief propagation function 388, 391, 395
Berkeley Data Analytics Stack 40, 205
betweenness 390-393, 397, 410, 433, 440-444

big data 2, 6, 9-13
big data analytics applications 41-46
big data characteristics 10
big data classification 7, 10, 13
big data store model 56
big data types 10, 12
BigTable 35, 90, 107, 108, 138, 147
BIRT 243
BLOB 95-97, 112, 130, 147, 189
Bloom filter 130, 350-352
broadcast function 218
broadcasting variable 233, 234
BSON 100, 124, 126, 127
bucket 3, 29, 97, 171, 180, 181, 198, 356, 359, 360
business intelligence 5, 20, 34, 39, 43, 212, 243, 247
business process 5, 10, 30, 34, 60, 212

C

candidate rules 286
CAP theorem 35, 56, 90-92, 118, 130
Cassandra 35, 36, 73, 85, 87, 90, 92, 117, 129-136, 147, 207, 212, 213, 240
Cassandra query language 130, 131, 133, 212, 213, 240
category variable 254, 304, 307, 325, 409, 419, 428, 463, 467, 468, 470, 471
centrality 390-394, 397, 403, 410, 433, 442, 443, 446
Ceph 208
chi-square 270, 311-313, 349

Chukwa 73
classification 78, 303-317, 325, 326, 329, 421-423, 477, 479
clique 396, 446, 447
closeness 294, 391-393, 397, 410, 433-443
cloud computing 15, 16
cloud resources 16, 17, 20
cloud services 27-29
cluster computing 17, 57, 59, 60
cluster of machines 16, 17, 35-38, 56-66, 68
clustering 78, 292-302, 305-307, 325, 326, 329, 330, 475
collaboration management 34
collaborative filtering 211, 280, 281, 291, 318-321, 323-327, 329, 394, 444
collating 160
collection data center 130
collection data type 123, 124, 127, 128, 171, 231, 416
collection frequency 427
collection map 143
collection, definition 147
column family 77, 103, 104, 106-108, 129, 131, 137, 145, 147
columnar data 103, 378
combining 88, 122, 354
command line 65, 133
command line interface 168, 172
command line utilities 58
composing MapReduce 160
composing Spark 236

concept extraction 412
conditional probability 315, 388-390, 448
conditional probability distribution 388, 389
conditional probability table 368, 389, 390, 395, 396
confidence level 349, 350
connected component 390, 445, 446
consistency 35, 57, 88, 91-93, 125, 132, 133, 147
container 71, 72, 123, 219
content-based filtering 322
correlation 263, 264, 267, 269, 278, 319, 324, 328, 370, 417, 428, 430, 442, 444
correlation coefficient 219, 263, 264, 370
cosine similarity 281-283, 320
COUGAR 340, 344
counting distinct 352, 353, 371
CPD *see* conditional probability distribution
CPT *see* conditional probability table
CQL *see* Cassandra query language
CQL (STREAM) *see* STREAM continuous query language
credit risk management 43, 44
cross correlation 161
CSV data format 3, 26-29, 31, 98, 192, 219, 229, 236-238, 459-461, 463, 464, 469-472, 475-480, 484, 485, 487, 492-493

D

data analytics 38-40
data architecture 18, 90, 115, 118
data architecture pattern 90, 118

data block 57-60, 63-65, 70, 80, 341, 401
data cleaning 25, 427
data consistency 57
data consumption 18, 19, 146
data cube 243, 431
data definition language 31, 93, 131, 174
data ingestion 19
data integration 33, 34
data management 13, 20, 32, 37, 341, 346, 347
data manipulation language 31, 176
data mart 5
data mining 240, 277, 285, 288, 290, 295, 303, 412, 413 , 417, 425, 426, 429
data model 3, 8, 35, 55, 77, 90, 92-95, 97, 118, 127, 131, 137, 138, 146, 171, 186, 189, 327, 339, 341, 409
data node 56, 65, 86, 89, 113, 117, 122, 136, 137, 146, 155, 233, 307, 341
data noise 24, 417
data pipeline 206, 211, 218, 221, 246
data pre-processing 25, 481
data replication *see* replication
data source 13, 21-23, 35-37, 44, 47, 64, 65, 185, 205, 208, 212, 214, 236, 240, 243, 246, 411, 412, 418, 453
data store 3, 13, 14, 20, 27-30, 33 , 35, 61, 63, 89, 92, 95, 97, 106, 107, 111, 113, 119, 123, 129, 137
data stream management system 341-344
data visualization 243, 244, 468
data warehouse 4, 33, 73, 78, 81, 198
database administration 4, 65, 76, 111
database connectivity 4, *see also* JDBC, ODBC

database definition 3

database maintenance 4

DataFrame 206, 215, 221, 397, 212, 214-216, 220, 221, 223, 230, 238, 239, 246, 453, 454, 461-466, 469, 470, 475, 476, 481, 483-485, 487-489

DB2 *see* IBM DB2

DBC driver 4

DDL *see* data definition language

decaying window 356, 362, 371

decision tree 234, 309-311, 314, 329, 419

degree distribution 443

descriptive analytics 6, 39

deserialization 108

deserializer 175, 206

dicing 221, 244

directed acyclic graph 74, 187, 205, 206

directed graph 206, 377, 378, 380, 388, 391-393, 403, 432, 449

directed multigraph 378, 398

distance measure 278, 280, 281, 284, 301, 302, 328

distance, definition 224, 273

distributed computing model 15, 47, 55, 56, 86

distributed database 32, 55, 91, 92, 112, 226, 403, 453

distributed datastore 3, 54, 63

distribution function 261, 378, *see also* probability distribution function

distribution model 119-121

DML *see* data manipulation language

document classification 412, 414, 419

document clustering 412

document collection 103
document data store 97, 100
document frequency 280, 414, 415, 427
document tree 97, 101, 102
DSMS *see* data stream management system
DStream 364-366
DynamoDB 92, 97

E

edit distance 281, 284
ego net *see* ego network
ego network 442
Enterprise server 27, 39, 69
entropy 313, 314
ETL 5, 20, 186, 199, 206, 213, 235, 236, 246
ETL pipeline 236
Euclidean distance 222, 223, 272-274, 281, 293, 296, 296, 298, 299, 328
EVAL function 195
event analytics 33
explanatory variable 254, 263

F

feature selection 417
feature variable 254
filter 193, 195, 196, 229, 230, 238, 351, 352, 366
see also, Bloom filter
filtering 25, 160, 224, 243 *see also* collaborative filtering, content-based filtering, knowledge-based Filtering

flat-file 3, 171, 228
Flink 61, 62
flow analysis 394
Flume 61, 73, 75
frequent itemset 234, 285-288, 325, 326, 329,
357-362, 372, 444
F-test 260, 262

G

Gini index 311
graph analytics 211, 376-386, 390, 392, 393,
395, 396
graph database 113-115, 137, 378, 381-383, 400, 403
graph edge 115, 161, 377, 378, 380, 381, 383, 384, 388, 389, 394, 395, 398-400,
409, 433, 445
graph model 113, 338, 377-380, 384, 386-388 , 393, 395, 400
graph node 114, 115, 377-384
graph partitioning 396, 399
graph processing 161, 364, 382, 399, 402
grid computing 17
GroupBy 152, 167, 178, 183-185, 221, 223, 229, 246, 462, 469
GVUDF 206, 223

H

Hadoop 37, 38, 57-59
Hadoop Common 59, 227
Hadoop distributed file-system 37, 59-62, 65-67, 107, 154, 208, 214, 233, 341,
455, 456, 468
Hadoop ecosystem 39, 58-62, 73, 80

Hadoop pipes 62
Hadoop streaming 62
hash function 97, 180, 284, 350-354 *see also* hash-key
hash map 283, 284, 350
hash ring 122
hash table 97
hash tree 286
hash-key 3, 86, 97, 106, 107
HBase 61, 73, 76, 77, 90, 92
HDFS *see* Hadoop Distributed File System
HDFS shell command 67
hierachal clustering 300, 301
HITS algorithm 438, 439
Hive 61, 65, 73, 78, 108, 109, 145, 167, 178, 182, 187, 198, 213, 214, 454, 463, 466-468
Hive data units 171
Hive file format 171
Hive server 168, 213, 214
HiveContext 216, 218, 222, 232
HiveQL 174-177, 181-183, 198, 216
horizontal scalability 15, 57
hub 438-441, 448
hyperplane 316, 421-423
hypothesis 234, 261, 262, 315

I

IaaS 16

IBM BigInsight 16, 76
IBM DB2 4
IBM G-System 397
IBM i data-library 22
in-degree 378, 392, 393, 399, 432-434, 437, 438, 443
influence propagation function 388
in-memory column format 32, 103, 104, 106, 107, 206, 211
in-memory graph computing 401
in-memory processing 5, 205, 206
in-memory row format 33, 103, 108
inner join 89, 148, 164, 466, 477
interaction detector 312
intersection 163, 280

J

Jaccard distance 281
Jaccard similarity 279-281, 291
JDBC 135, 168, 170, 213, 214
job 56
job scheduling 56
JobTracker 65, 66, 68, 69, 136, 150, 158
join 88, 89, 92, 148, 164, 181-183, 194, 216, 365, 382, 400, 465-467, 476, 478, 481, 485, 488
JSON 31, 90, 98-103, 123, 214-216, 218, 227, 237

K

Kafka 40, 364
kernel function 260, 261, 422

keyspace 131-133, 148
key-value pair 3, 95, 137, 150-152, 156-157, 166, 190
K-mean 295-300, 326, 475, 477
K-Nearest-Neighbour 272-274, 307, 328, 419, 444
K-Nearest-Neighbour search 276-278
KNN *see* K-Nearest Neighbour
K-NNS *see* K-Nearest-Neighbour search
knowledge discovery 240, 288, 396, 412
Knowledge-based filtering 322

L

lambda architecture 341
left outer-Join 183, 194, 466
link analysis 115, 431, 432
local partitioning *see* partitioning

M

machine learning 39, 73, 78, 234, 235, 252, 474
Mahout *see* Apache Mahout
Manhattan distance 273, 274, 279, 293
Mapper 67-70, 151-153, 156, 158, 160-166, 461, 462
MapReduce 38, 58, 67-69
market basket model 288, 289, 329
Markov network 387, 388, 390, 403
massively parallel processing 15
master-slave 120-122
matplotlib 244, 468-488

Mesos 37, 208, 209
metadata 33, 92, 93, 107, 111, 172
metastore 168, 169, 207
ML pipeline 234, 235
MLib 40, 211, 234, 235
moment 259, 261, 263, 324, 355
MongoDB 90, 92, 100, 121, 123-128, 138
multi structured data 8
multi-dimensional data cube 243
multihash algorithm 360, 372
multi-master DB 138
multiple regression 270, 271
multistage algorithm 359, 360, 372
multivariate distribution 267
multivariate neighbours 276
multivariate regression 314

N

Naïve Bayes classifier 315, 419, 420, 448,
479-481, 483
name-value pair 3, 36
natural Join 148, 164-166
Neo4j 115, 383, 384
nested table 206, 238
node manager 71, 72
NoSQL 35-38, 88-95, 115-119, 122, 146, 147
null hypothesis 262

NumPy 218, 219, 234, 235, 467, 473-475, 477, 478, 480, 488

O

object data store 111, 147, 214

objective function 270, 309

ODBC 23, 168, 213

OLAP 33, 106, 212-213, 240

OLTP 33, 103, 213

online analytical Processing *see* OLAP

online transaction processing *see* OLTP

Oozie 73, 74

ORC 109, 110, 171, 214

outcome variable 254

out-degree 378, 392, 399, 432, 433, 443

outer-join 183, 194, 466

outlier 24, 25, 224, 254, 257, 258, 261, 417

P

PaaS 16

PageRank 395, 399-401, 432-437, 440, 443, 448

Pandas 205, 220-223, 246, 464-466

Panel 220

parallel tasks 15

Parquet 110, 111, 206, 214, 215, 339

parsing 160, 414

partition 171

partitioning 152, 178, 179

part-of-speech tagging 413

pattern analysis 431, 429
pattern discovery 429-431
pdf *see* probability distribution function
Pearson correlation 264
peer-to-peer 121
Pig 73, 78, 81, 145, 185-195
Pig data type 189
Pig Latin 78, 185-195
Piggy Bank 195
population 257
predictive analytics 6, 39, 241
predictor variable 254, 270, 272, 483
pre-processing 29, 38 *see also* data pre-processing
prescriptive analytics 6, 39
primary master 65
probability distribution function 259, 387
process 5
process matrix 5
projection 148, 163, 193
PySpark 38, 212, 213, 454, 218
Python 146, 205, 208, 217, 218, 226, 246

Q

query language 102, 344

R

Rack 56, 63, 64, 80

Random Forest 307, 316, 326, 329
RCFile *see* record columnar file
RDBMS 4, 32, 38, 48, 88, 89
RDD 209, 211, 246, 366, 397, 398
real-time analytics 363, 372
real-time analytics platform 363
real-time processing 5, 205, 346, 347
recommender 318, 321, 322, 327, 329, 394
record columnar file 108
RecordReader 151, 154
Reducer 67, 69, 153, 157, 158, 160-166, 461. 462
regression model 267, 314, 328, 483-492
relation 92, 190, 193, 194
relational database management system *see* RDBMS
relational operator 192-195
relationship 115, 119, 124, 161, 254-256, 262, 263, 267, 328, 377-381, 383, 403
replication 107, 117, 120-122, 124, 130, 132
report-designer 242
report-engine 243
resilient distributed-dataset *see* RDD
resource 57
resource manager 119
resource scheduling 69
response variable 254, 267, 311
right outer-join 183, 194, 466
row-based data 103
RTAP *see* real-time analytics platform

RTAP applications 368

S

S3 112, 209

SaaS 17

sample 194, 257

scalability 8, 14, 47, 58, 86

scalable 58, 60, 307

scatter plot 255, 256, 471

schema-less 92, 93, 137

SchemaRDD 206, 214, 230

SciPy 218, 219, 487

secondary master 65

semi-structured data 8, 31, 48, 90

sequence File 78, 227

SerDe 206

serializer function 206

server definition 6, 38

service oriented architecture 6

Services definition 6

sharding 86, 89, 120, 125

shared nothing 85, 119, 307

shell command *see* HDFS shell command

shuffle and sort 152

similar items 277, 328

similarity 277, 280, 281, 291

similarity coefficient 281

SimRank 445
singular value decomposition 324
slave node 65, 120
slicing 221
SOA *see* service oriented architecture
social network 441, 449, 409
sorting 160
spam detection 442
Spark *see* Apache Spark
Spark ecosystem 364
Spark SQL 212, 213, 246, 453
Spark Streaming 211, 364
SparkContext 203, 215, 227
SparkContext 227, 465
split 194
spreadsheet 3
SQL 4, 31
SQLContext 215, 216, 218
SQL-like script 146
standard deviation 258
standard error of estimate 258
statistical significance 271
statistical technique 430
stream analytics 4, 363
STREAM continuous query language 344
structured data 7
structured data sources 21

support vector 421
support vector machine 316, 421, 477
SVD *see* singular value decomposition
SVM classifier *see* support vector machine

T

table 3, 32
table partitioning 178
Tableau 244
term frequency 414-416
text analytics process pipeline 412, 413
text cleanup 413
text features generation 414
text mining 409, 411, 448
text pre-processing 413
TF-IDF 414
Top K shortest paths 394
transaction 4
transform 5, 228, 235, 246
triangles count 445
triangles count Algorithm 399, 446
tuple 88, 147, 189

U

UDF *see* user-defined function
unstructured data 7, 9
unstructured data sources 23
user-defined function 195, 199, 206, 221, 466, 467

utility based cloud 58

V

vector space model 414-415

Vectorized UDF 206, 222

vertical scalability 14, 57

View 88, 181

VUDF *see* vectorized UDF

W

web community 440, 448

web content mining 410, 425, 426

web data 7, 424, 448

web graph 426

web mining 425, 448

web structure 432, 449

web structure mining 426

web usage mining 425, 426, 428, 429, 448

X

XML 100-103

XPath 102

Y

YARN 59-61, 71, 80, 146

Z

ZooKeeper 61, 65, 66, 74, 80, 364