Big Data Analytics with Spark and Python

Course goal

- Train students to kick start their journey with Big Data Analytics with Apache Spark with simple explanations and easy to do exercises

Why learn Apache Spark?

- Hottest tool /product in the Big Data Analytics field.
- Used by more and more companies for Analytics and Machine Learning
- Hadoop/Map Reduce applications migrating to Spark
- More and more third party support
- Huge current and forecasted demand for skilled professionals

What you achieve by taking this course

- Understand the concepts and life cycle of Data Science and Analytics
- Develop proficiency to use Apache Spark for all stages of analytics
- Learn Data Engineering tools and techniques with Spark
- Acquire knowledge of different machine learning techniques and know when and how to use them.
- Become a full-fledged Big Data Analyst who can immediately contribute to real-life Analytics projects

Course Structure

- Hadoop and Spark Concepts
- Spark Programming including Spark SQL and Spark Streaming
- Basics of Real Time Data Science
- Machine Learning with Spark
- End-to-End use cases
- Resource Bundle

Things not covered

- Python basics
- Elaborate coverage of the Spark library
- Spark Cluster setup and administration

Guidelines to students

- Machine Learning and Data Science is a complex subject. Needs significant efforts to understand it.
  - Review and re-review videos and exercises
  - Seek out other help – books, online documentations, support forums
- If you have queries, doubts or concerns, please send a private message or post a discussion question
  - We would be happy to address them as soon as possible
- We are constantly improving our courses so all feedback is welcome
  - Feedback through private messages / emails.
- At the end of the course, if you like it, please leave a review

Relationship with other V2 Maestros courses

- Our courses are focused on Data Science related topics
  - Technologies
  - Processes
  - Tools and Techniques
- We focus on making our courses self sufficient
- If you are an existing V2 Maestros student, you will see some content and examples repeated across courses

We hope this course helps you to advance your career.
Best of luck !

Hadoop Technologies

Overview

Big Data & Hadoop Overview

What is Big Data?

- Broad general term for data sets so large and complex that traditional data processing and storage techniques are inadequate
  - Traditional RDBMS and business applications
- Volume (TB, PB)
- Variety (web, photo, video, audio, unstructured data, mobile, social)
- Velocity (batch, periodic, real time)
- Veracity (quality of data – dirty)

Why Big Data

- Web and Cloud applications created the need to store and process huge amounts of data
- Traditional RDBMSs do not fit the role
  - Only good for numbers, structured and clean data
  - Scaling required very expensive hardware
  - Fault tolerance was again expensive
- Existing processing techniques cannot scale without extensive code development

Evolution of Big Data (Hadoop)

- 2002 - Doug Cutting and Mike Cafarella start working on Nutch
- 2003 – Google publishes GFS & MapReduce
- 2004 – Doug Cutting adds GFS & MR to Nutch
- 2006 – Yahoo hires Doug Cutting and Hadoop is created
- 2008 – Applications of Hadoop start to emerge
- 2009 – New companies which built on Hadoop start to emerge
- Hadoop and its eco-system starts to grow and expand

What is Hadoop ?

- Doug Cutting named his product as "Hadoop" based on the name of a an elephant toy which his kid named as Hadoop
- The Hadoop product consist of 2 components
  - Hadoop Distributed File System (HDFS)
  - Map Reduce Programming Paradigm
- Hadoop forms a "platform" on which a number of applications are built.
  - Data Ingestion, Processing and Analytics

Things about Hadoop

- Unix based (no windows support)
- Built using Java
- Not much UI. Most actions are command line based.

V2 Maestros
The Data Science Experts

Setting up your Hadoop Environment

If you already have a Hadoop setup, you can skip this section.

Cloudera QuickStart VM

- Single box installation containing running instances all Hadoop components
- Linux based. (! ☹). Linux familiarity is a pre-requisite.
- Hadoop in general is not that user friendly (for folks used to windows)
- Minimum 4 GB. Need 8 GB for good response times.
- Can be installed as a VM on windows
- Downloads & setup instructions available at
  - http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cloudera_quickstart_vm.html

HDFS

Features of HDFS

- Another "Distributed File System"
  - Files and Directories.
- Optimized for very large files (TB, PB)
- Optimized for write-once, read-many
- Fault-tolerance by default. No backups required.
- Data replication happens all the time.
- Runs on commodity hardware
- Moves code to where data resides

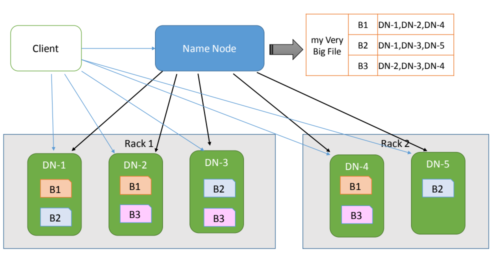
HDFS Architecture

- Master Slave architecture
- Built as a HDFS Cluster. Each cluster has one to many nodes.
- One "NameNode" per cluster (Master)
  - Master who manages the cluster
  - Maintains meta-data about the entire cluster
  - Allocates work to data nodes
- One "DataNode" per node (Slaves)
  - Storage, and read-write operations

Storing files in HDFS

- Files split up as data blocks (64 MB size by default)
- Each block is replicated across multiple data nodes (3 copies)
- NameNode maintains list of blocks that makes up a file.
- File writes (only create, no update)
  - Client contacts NameNode for destination to write
  - NameNode provides list of DataNodes to write to.
  - Client writes directly to the DataNode.
- File reads
  - Client contacts NameNode for list of blocks and locations
  - NameNode returns the list
  - Client reads directly from the DataNodes
- HDFS Storage is "rack-aware"

HDFS Architecture



Map Reduce

Map Reduce Overview

- A new programming paradigm built to exploit the parallel nature of HDFS data.
- Batch mode execution.
- Moves program code to data nodes.
- Multiple Map Reduce jobs can be chained to create a larger solution
- Designing jobs require thinking in Map Reduce paradigm

Map Reduce components

- Map Reduce jobs are executed using the Job Manager and the Task Manager.
- Job Manager runs on the NameNode.
  - "Plans" the execution of the job on the task managers
  - Works with the NameNode.
  - Returns results to client
- Task Managers run on each of the Data Nodes
  - Executes the map and reduce functions.
- In YARN (Map Reduce v2), the function of the job manager is split between the Applications Manager and the Resource Manager.

What is Map and Reduce

- Map
  - A Function (a program)
  - Works on 1 line of the file at a time
  - Output is keys and values.
- Reduce
  - A function
  - Works on one key at a time.
  - Output is key and values.

How it works - Input

- Map Function
- Reduce Function
- Files containing the map and reduce functions
- Input HDFS directory
- Output HDFS directory

How it works - Splits

- The input data is split into "splits". A split is a copy of contiguous HDFS blocks of data in the input file.
- Each split exists on a specific data node.
- The client (client library/ command line) copies the files containing the map function to each data node identified.
- The data node should contain execution capabilities for the code file.
- The local Task Manager process then executes the map function. Multiple such processes will run in parallel on different splits

How it works – Map function

- The Task Manager iterates over each line in the input split and passes it to the map function.
- The "map" function is called for each input line.
- The line is of "Text" format
- It is the responsibility of the Map function to interpret / split / convert / process the line.
- Typical functionality of Map functions include Data Cleansing and Filtering
- Map function should not work "across" lines.
- Map outputs key-value pairs as output.
- Each run can output the same key multiple times.

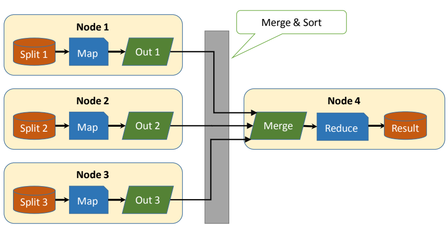
How it works – merge / sort

- Sort and merge is done by Hadoop
- The outputs of all map executions from different DataNodes are merged.
- This merged data is sorted by the keys
- Values for the same key are then converted to a list.
- This <key,value list> then becomes the input for the reduce function.

How it works – Reduce function

- Typically there is only one reduce execution.
- Input is the <key, value list> from the sort/merge operation. It is iterated key by key.
- Reduce function called once for each key.
- Typical usage is summarization, analysis, joins etc.
- Reduce functions can work across multiple keys.
- Multiple reduce executions can be used if operations are limited to keys. Data is split by keys between multiple reduce instances.
- Output of the reduce function is placed in the output directory.

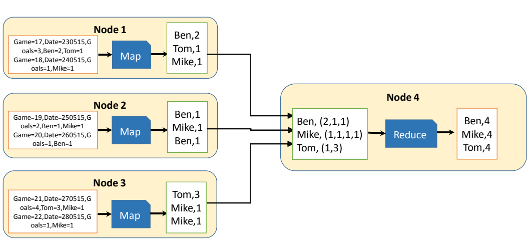
Map Reduce Execution



Map Reduce Example

- Input : Score sheet of soccer games for a team
  - Game=17,Date=230515,Goals=3,Ben=2,Tom=1
  - Game=18,Date=240515,Goals=1,Mike=1
  - Game=19,Date=250515,Goals=2,Ben=1,Mike=1
  - Game=20,Date=260515,Goals=1,Ben=1
  - Game=21,Date=270515,Goals=4,Tom=3,Mike=1
  - Game=22,Date=280515,Goals=1,Mike=1
- Program output : Total goals by player

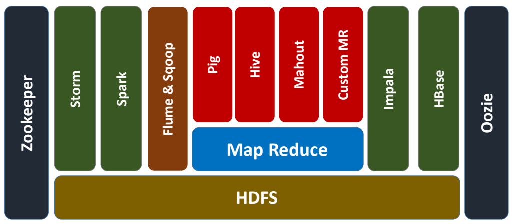
Example Program Flow



V2 Maestros
The Data Science Experts

Hadoop Stack

Hadoop Technology Stack



V2 Maestros
The Data Science Experts

Apache Spark

V2 Maestros
The Data Science Experts

Introduction to Spark

What is Apache Spark

http://spark.apache.org/

- A fast and general engine for large-scale data processing
- A Open-source cluster computing framework
- End-to-End Analytics platform
- Developed to overcome limitations of Hadoop/Map Reduce
- Runs from a single desktop or a huge cluster
- Iterative, interactive or stream processing
- Supports multiple languages – Scala, Python, R, Java
- Major companies like Amazon, eBay, Yahoo use Spark.

Advantages of Spark

- A fast-growing Open Source engine
- Many times faster than map-reduce
  - Keeps data in memory
- Runs alongside other Hadoop components
- Support for many programming languages
  - Scala, R, python, Java, piping
  - Same functionality across multiple languages
- Multiple options and libraries – Graph, SQL, ML, Streaming
- Workings with multiple management frameworks

Spark Use Cases

- Data Integration and ETL
- Interactive Analytics
- High Performance Batch computation
- Machine Learning and Advanced Analytics
- Real time stream processing
- Example applications
  - Credit Card Fraud Detection
  - Network Intrusion Detection
  - Advertisement Targeting

Typical Spark workflow

- Load data from source
  - HDFS, NoSQL,S3, real time sources
- Transform Data
  - Filter, Clean, Join, Enhance
- Store processed data
  - Memory, HDFS, NoSQL
- Interactive Analytics
  - Shells, Spark SQL, third-party tools
- Machine Learning
- Action

Online Reference

- http://spark.apache.org

V2 Maestros
The Data Science Experts

Spark Architecture

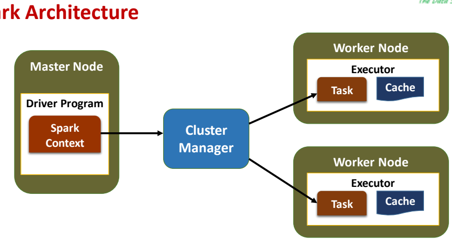
Spark Framework

Programming	Scala	Python	R	Java	Tools
Library	Spark SQL	ML Lib	GraphX	Streaming	
Engine	Spark Core				
Management	YARN	Mesos	Spark Scheduler		
Storage	Local	HDFS	S3	RDBMS	NoSQL

Resilient Distributed Datasets (RDD)

- Spark is built around RDDs. You create, transform, analyze and store RDDs in a Spark program.
- The Dataset contains a collection of elements of any type.
  - Strings, Lines, rows, objects, collections
- The Dataset can be partitioned and distributed across multiple nodes
- RDDs are immutable. They cant be changed.
- They can be cached and persisted
- Transformations act on RDDs to create a new RDD
- Actions analyze RDDs to provide a result

Spark Architecture



Spark scalability

- Single JVM
  - Runs on a single box (Linux or Windows)
  - All components (Driver, executors) run within the same JVM
- Managed Cluster
  - Can scale from 2 to thousands of nodes
  - Can use any cluster manager for managing nodes
  - Data is distributed and processed on all nodes

Driver Program

- The main executable program from where Spark operations are performed
- Controls and co-ordinates all operations
- The Driver program is the “main” class.
- Executes parallel operations on a cluster
- Defines RDDs
- Each driver program execution is a “Job”

SparkContext

- Driver accesses Spark functionality through a SparkContext object.
- Represents a connection to the computing cluster
- Used to build RDDs.
- Works with the cluster manager
- Manages executors running on Worker nodes
- Splits jobs as parallel “tasks” and executes them on worker nodes
- Partitions RDDs and distributes them on the cluster
- Collects results and presents them to the Driver Program

Spark modes

- Batch mode
  - A program is scheduled for execution through the scheduler
  - Runs fully at periodic intervals and processes data
- Interactive mode
  - An interactive shell is used by the user to execute Spark commands one-by-one.
  - Shell acts as the Driver program and provides SparkContext
  - Can run tasks on a cluster
- Streaming mode
  - An always running program continuously processes data as it arrives

Lazy evaluation

- Lazy evaluation means Spark will not load or transform data unless an action is performed
  - Load file into RDD
  - Filter the RDD
  - Count no. of elements (only now loading and filtering happens)
- Helps internally optimize operations and resource usage
- Life easy for developers – can write chaining operations
- Watch out during troubleshooting – errors found while executing actions might be related to earlier transformations

Transformations

Overview

- Perform operation on one RDD and create a new RDD
- Operate on one element at a time
- Lazy evaluation
- Can be distributed across multiple nodes based on the partitions they act upon

Map

```
newRdd=rdd.map(function)
```

- Works similar to the Map Reduce "Map"
- Act upon each element and perform some operation
  - Element level computation or transformation
- Result RDD may have the same number of elements as original RDD
- Result can be of different type
- Can pass functions to this operation to perform complex tasks
- Use Cases
  - Data Standardization – First Name, Last Name
  - Element level computations – compute tax
  - Add new attributes – Grades based on test scores

flatMap

```
newRdd=rdd.flatMap(function)
```

- Works the same way as map
- Can return more elements than the original map
- Use to break up elements in the original map and create a new map
  - Split strings in the original map
  - Extract child elements from a nested json string

Filter

```
newRdd=rdd.filter(function)
```

- Filter a RDD to select elements that match a condition
- Result RDD smaller than the original RDD
- A function can be passed as a condition to perform complex filtering
  - Returns a true/false

Set Operations

- Set operations are performed on two RDDs
- Union – Return a new dataset that contains the union of the elements in the source dataset and the argument.
  - `unionRDD=firstRDD.union(secondRDD)`
- Intersection - Return a new RDD that contains the intersection of elements in the source dataset and the argument.
  - `intersectionRDD=firstRDD.intersect(secondRDD)`

Pair RDDs

- Pair RDDs are a special type of RDDs that can store key value pairs.
- All transformations for regular RDDs available for Pair RDDs
- Spark supports a set of special functions to handle Pair RDD operations
  - mapValues : transform each value without changing the key
  - flatMapValues : generate multiple values with the same key

Actions

Introduction to actions

- Act on a RDD and product a result (not a RDD)
- Lazy evaluation – Spark does not act until it sees an action
- Simple actions
  - `collect` – return all elements in the RDD as an array. Use to trigger execution or print values
  - `count` – count the number of elements in the RDD
  - `first` – returns the first element in the RDD
  - `take(n)` – returns the first n elements

reduce

- Perform an operation across all elements of an RDD
  - sum, count etc.
- The operation is a function that takes as input two values.
- The function is called for every element in the RDD

```
inputRDD = [ a, b, c, d, e ]   and the function is  func(x,y)
func( func( func( func(a,b), c), d), e)
```

- Example

```
vals = [3,5,2,4,1]
sum(x,y) { return x + y }
sum( sum( sum( sum(3,5), 2), 4), 1) = 15
```


aggregate

- Perform parallel computations on partitions and combine them
- A Sequence operation happens on each partition
- A Combine operation helps combine the results
- Can do multiple computations at the same time.
- Takes a initial value for each operation – it should be an identity value
  - `Rdd=[3,5,4,7,4]`
  - `seqOp = (lambda x, y: (x[0]+y, x[1]*y))`
  - `combOp = (lambda x, y: (x[0]+y[0], x[1]*y[1]))`
  - `collData.aggregate((0,1), seqOp, combOp)`

If there are 2 partitions

```
Rdd1=[3,5,4]  Rdd2[7,4]
Seqeunce operation will produce [(12,60),(11,28)]
Combine operation will produce (23,1680)
```


Pair RDD Actions

- countByKey – produces a count by each key in the RDD
- groupByKey – perform aggregation like sum, average by key
- reduceByKey – perform reduce, but by key
- aggregateByKey – perform aggregate by key
- Join - join multiple RDDs with the same key

Loading and Storing Data

Creating RDDs

- RDDs can be created from a number of sources
  - Text Files
  - JSON
  - Parallelize() on collections
  - Sequence files

Storing RDDs

- Spark provides simple functions to persist RDDs to a variety of data sinks
  - Text Files
  - JSON
  - Sequence Files
  - Collections
- For optimization use language specific libraries for persistence than using Spark utilities.

Partitioning and Persistence

Partitioning

- By default all RDDs are partitioned
  - spark.default.parallelism parameter
  - Default is the total no. of cores available across the entire cluster
- Should configure for large clusters
- Can be specified during RDD creation explicitly
- Derived RDD take the same number as the source.

Persistence

- By default, Spark loads an RDD whenever it required. It drops it once the action is over
  - It will load and re-compute the RDD chain, each time a different operation is performed
- Persistence allows the intermediate RDD to be persisted so it need not have to be recomputed.
- persist() can persist the RDD in memory, disk, shared or in other third party sinks
- cache() provides the default persist() – in memory

Advanced Spark

Broadcast variables

- A read-only variable that is shared by all nodes
- Used for lookup tables or similar functions
- Spark optimizes distribution and storage for better performance.

Accumulators

- A shared variable across nodes that can be updated by each node
- Helps compute items not done through reduce operations
- Spark optimizes distribution and takes care of race conditions

Spark SQL

Overview

- A library built on Spark Core that supports SQL like data and operations
- Make it easy for traditional RDBMS developers to transition to big data
- Works with "structured" data that has a schema
- Seamlessly mix SQL queries with Spark programs.
- Supports JDBC
- Helps "mix" n "match" different RDBMS and NoSQL Data sources

DataFrame

- A distributed collection of data organized as rows and columns
- Has a schema – column names, data types
- Built upon RDD, Spark optimizes better since it knows the schema
- Can be created from and persisted to a variety of sources
  - CSV
  - Database tables
  - Hive / NoSQL tables
  - JSON
  - RDD

Operations supported by Data Frames

- `filter` – filter data based on a condition
- `join` – join two Data Frames based on common column
- `groupBy` – group data frames by specific column values
- `agg` – compute aggregates like sum, average.
- `registerAsTempTable` – register the Data Frame as a table within SQLContext
- Operations can be nested.

SQLContext

- All functionality for Spark SQL accessed through a SQLContext
- Derived from SparkContext
- Data Frames are created through SQLContext
- Provides a standard interface to work across different data sources
- Can register Data Frames as temp table and then run SQL queries on them

V2 Maestros
The Data Science Experts

Spark Streaming

Overview

- Typically, Analytics is performed on data at rest
  - Databases, flat files etc.
- Some use cases require real time analytics
  - Fraud detection, click stream processing
- Spark Streaming is built for this purpose
- Spark Streaming helps you to
  - Look at data as they are created/arrive from source
  - Transform, summarize, analyze
  - Perform machine learning
  - Predict in real time

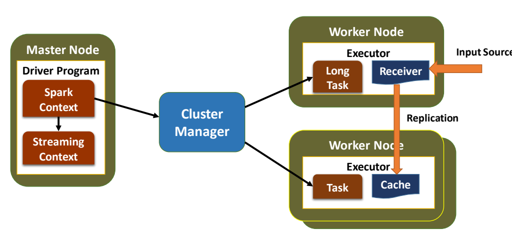
Use Cases

- Credit Card Fraud Detection
- Spam Filtering
- Network Intrusion Detection
- Real time social media analytics
- Click Stream analytics and recommendations
- Ad recommendations
- Stock Market analytics

Spark Streaming sources

- Flat files (as they are created)
- TCP/IP
- Apache Flume
- Apache Kafka
- Amazon Kinesis
- Twitter, Facebook and other social media

Spark Streaming Architecture



DStreams

- A Streaming context is created from the Spark context to enable streaming
- Streaming creates a DStream (Discretized Stream) on which processing occurs
- A micro-batch window is setup for the DStream
- Data is received, accumulated as a micro-batch and processed as a micro-batch
- Each micro-batch is an RDD
- Regular RDD operations can be applied on the DStream RDD

DStream processing

- Spark collects incoming data for each interval (micro-batch)
- Data is collected as an RDD for that interval.
- It then calls all transformations and operations that applies for that DStream or derived DStreams
- Global variables can be used to track data across DStreams
- Windowing functions are available for computing across multiple DStreams.
  - Window size multiple of interval
  - Sliding interval multiple of interval

V2 Maestros
The Data Science Experts

What is Data Science

Understanding the domain

V2 Maestros
The Data Science Experts

Definitions

Across the web

Data Science

- Skill of extracting of knowledge from data
- Using knowledge to predict the unknown
- Improve business outcomes with the power of data
- Employ techniques and theories drawn from broad areas of mathematics, statistics and information technology

Data Scientist

- A practitioner of data science
- Expertise in data engineering, analytics, statistics and business domain
- Investigate complex business problems and use data to provide solutions

Data

The foundation of Data Science

Entity

- A thing that exists about which we research and predict in data science.
- Entity has a business context.
- Customer of a business
- Patient at a hospital. The same person can be a patient and a customer, but the business context is different.
- Car. Entities can be non living things

Characteristics

- Every entity has a set of characteristics. These are unique properties
- Properties too have a business context
- Customer : Age, income group, gender, education
- Patient: Age, Blood Pressure, Weight, Family history.
- Car: Make, Model, Year, Engine, VIN

Environment

- Environment points to the eco-system in which the entity exists or functions.
- Environment is shared among entities. Multiple entities belong to the same environment
- Environment affects an entity's behavior
- Customer : Country, City, Work Place
- Patient: City, Climate .
- Car: Use (City/highway), Climate

Event

- A significant business activity in which an entity participates.
- Events happen in a said environment.
- Customer : Browsing, store visit, sales call
- Patient: Doctor visit, blood test
- Car: Smog test, comparison test

Behavior

- What an entity does during an event.
- Entities may have different behaviors in different environments
- Customer : Phone Call vs email, Clickstream, response to offers
- Patient: Nausea, light-headed, cramps
- Car: Skid, acceleration, stopping distances

Outcome

- The result of an activity deemed significant by the business.
- Outcome values can be
  - Boolean (Yes/No, Pass/Fail)
  - Continuous (a numeric value)
  - Class (identification of type)
- Customer : Sale (Boolean), sale value (continuous)
- Patient: Blood Pressure value (continuous). Diabetes type (class)
- Car: Smog levels (class), stopping distances (continuous), smog passed (Boolean), car type (class)

Observation

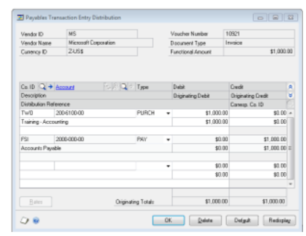
- A measurement of an event deemed significant by the business.
- Captures information about
  - Entities involved
  - Characteristics of the entities
  - Behavior
  - Environment in which the behavior happens
  - outcomes
- An observation is also called a system of record
- Customer : A phone call record, a buying transaction, an email offer
- Patient: A doctor visit record, a test result, a data capture from a monitoring device
- Car: Service record, smog test result

Dataset

- A collection of observations
- Each observation is typically called a record
- Each record has a set of <u>attributes</u> that point to characteristics, behavior or outcomes.
- A dataset can be
  - Structured (database records, spreadsheet)
  - Unstructured (twitter feeds, news paper articles)
  - Semi-structured (email)
- Data scientists collect and work on datasets to learn about entities and predict their future behavior/ outcomes.

Structured Data

- Attributes are labeled and distinctly visible.
- Easily searchable and query able.
- Stored easily in tables



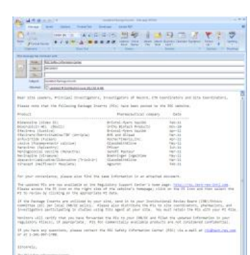
Unstructured Data

- Data is continuous text
- Attributes are not distinctly labeled. They are present within the data.
- Querying is not easy.

The Mazda3 is on a very short list of compact cars that are available as a hatchback or a sedan. It also comes with two 6-speed transmissions -- manual or automatic -- and choice of two 4-cylinder engines -- a 155-horsepower 2.0-liter or a 184-horsepower 2.5-liter -- and all of those variations are available with either body style. Its best fuel economy is an EPA-rated 41 mpg on the highway, which is near the top of the class for gasoline-powered cars (tying the Honda Civic, yet another trait they share). That rating applies to the 2.0-liter engine, whether it's backed by a manual or automatic transmission.

Semi-structured Data

- Mix of structured and unstructured.
- Some attributes are distinctly labeled. Others are hidden within free text



Summary

- Entity
- Characteristics
- Environment
- Event
- Behavior
- Outcomes
- Observation
- Dataset

V2 Maestros
The Data Science Experts

Learning

Discovering knowledge from Data

Relationships

- Attributes in a dataset exhibit relationships
- Relationships “model” the real world and have a logical “explanation”
- For attributes A and B the relationships can be
  - When A occurs, B also occurs
  - When A occurs B does not occur
  - When A increases B also increases
  - When A increases B decreases
- Relationships can involve multiple attributes too
  - When A is present and B increases, C will decrease

Relationships - Examples

- Customer
  - As age goes up, spending capacity goes up. (AGE and REVENUE)
  - Urban customers buy more internet bandwidth (LOCATION and BANDWIDTH)
- Patient
  - Older patients have more prevalence of Diabetes (AGE and DISEASE LEVEL)
  - Overweight patients typically have higher cholesterol levels (WEIGHT and HDL)
- Car
  - The more cylinders a car has, the mileage tends to be lower (CYLINDERS and MILEAGE)
  - Sports Cars have more insurance rates (TYPE and RATES)

Relationships

- Consistent vs Incidental Patterns in Data
- Correlations
- Signals and noise

What is Learning

- Learning implies learning about relationships.
- It involves
  - Taking a domain
  - Understanding the attributes that represent the domain
  - Collecting data
  - Understanding relationships between the attributes
- Model is the outcome of learning

Model

- A simplified, approximated representation of a real world phenomenon
- Captures key attributes and their relationships
- Mathematical model – represents relationships as an equation
- Blood Pressure

    BP = 56 + (AGE * .8) + (WEIGHT * .14) + (LDL * .009)
- Decision Tree model – represents the outcome as a decision tree
- Buying a music CD

    If AGE < 25 and GENDER=MALE, buy BEYONCE-CD = YES
- Accuracy of models depends on strength of relationships between attributes

Prediction

- A model can be used to predict unknown attributes

    BP = 56 + (AGE * .8) + (WEIGHT * .14) + (LDL * .009)
- The above model represents the relationships between BP, AGE, WEIGHT and LDL.
- If 3 of the 4 attributes are known, the model can be used to predict the 4^{th}.
- The above equation can be considered the prediction algorithm
- Relationships can be a lot more complex, leading to complex models and prediction algorithms.

Predictors and outcomes

- Outcomes are attributes that you want to predict
- Predictors are attributes that are used to predict outcomes.
- Learning is all about building models that can be used to predict outcomes (outputs) using the predictors (inputs)

Example	Predictors	Outcomes
Customer	Age, Income Range, Location	Buy? Yes/No
Patient	Age, Blood Pressure, Weight	Diabetic?
Car	Cylinders, acceleration	Sports vs family

Humans vs machines

- Humans understand relationships and predict all the time.
- Build humans can only handle finite amount of data
    - One shop keeper can know preferences of 100 customers, not 10 million of them
- Machines (computers) come into play when the number of entities and data about them are large
- There in comes machine learning, predictive analytics and data science

So what is Data Science ?

- Picking a problem in a specified domain
- Understanding the problem domain (entities and attributes)
- Collect datasets that represent the entities
- Discover relationships (Learning)
    - When computers are used for this purpose, its called machine learning.
- Build models that represent relationships
    - Uses past data where all predictors and outcomes are known
- Use models for predicting outcomes
    - Current/ future data – predictors known, outcomes unknown

Data Science Example – Website Shopper

- Problem : Predict if the shopper will buy a smartphone
- Data: Past purchase history of shoppers
    - Shopper characteristics (age, gender, income etc.)
    - Seasonal information
    - Others..
- Build Model
    - Decision model based on shopper and seasonal entities
    - Built every week
- Prediction
    - When a new shopper is browsing, predict if the shopper will buy
- Action
    - Offer Chat help

Real Time Data Science

What is real time?

- Faster processing
- Real time Analytics
- Real time model building
- Real time predictions.

Real time processing with Spark

- Connectors to real time streaming sources
- Data can be cleansed and transformed in stream.
- Stored in in-memory data stores –
- Reporting/Analytics using Spark SQL or 3rd party tools

Real time model building

- Not a real time activity
- Models need a large corpus of data
  - Volume
  - Variety
- Manually evaluated
- Needs exploring different algorithms
- Needs tuning.
- Build once, use many

Real time predictions

- Receive, cleanse and transform data in real time
- Load a saved model and use it for prediction.
- Store predictions in in-memory DB and use it for real time actions

Splitting work between Language and Spark

- Use language
  - Receiving data
  - Storing data
  - Communicating with other systems
- Use Spark
  - Data cleansing
  - Transformation
  - Model building
  - Predictions

Analytics and Predictions

Types of Analytics

Types of Analytics

Type of Analytics	Description
Descriptive	Understand what happened
Exploratory	Find out why something is happening
Inferential	Understand a population from a sample
Predictive	Forecast what is going to happen
Causal	What happens to one variable when you change another
Deep	Use of advanced techniques to understand large and multi-source datasets

Exploratory Data Analysis

Goals of EDA

- Understand the predictors and targets in the data set
  - Spreads
  - Correlations
- Uncover the patterns and trends
- Find key variables and eliminate unwanted variables
- Detect outliers
- Validate previous data ingestion processes for possible mistakes
- Test assumptions and hypothesis

Tools used for EDA

- Correlation matrices
- Boxplots
- Scatterplots
- Principal component Analysis
- Histograms

Machine Learning

-

Overview

- Data contains attributes
- Attributes show relationships (correlation) between entities
- Learning – understanding relationships between entities
- Machine Learning – a computer analyzing the data and learning about relationships
- Machine Learning results in a model built using the data
- Models can be used for grouping and prediction

Data for machine learning

- Machines only understand numbers
- Text Data need to be converted to equivalent numerical representations for ML algorithms to work.
- Number representation
  - (Excellent, Good, Bad can be converted to 1,2,3)
- Boolean variables
  - 3 new Indicator variables called Rating-Excellent, Rating-Good, Rating-Bad with values 0/1
- Document Term matrix

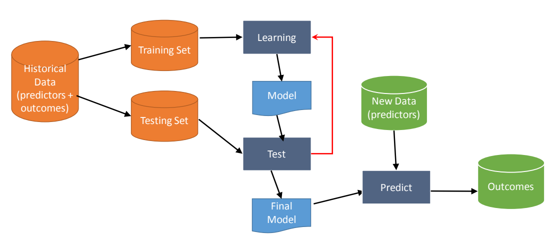
Unsupervised Learning

- Finding hidden structure / similarity / grouping in data
- Observations grouped based on similarity exhibited by entities
- Similarity between entities could be by
  - Distance between values
  - Presence / Absence
- Types
  - Clustering
  - Association Rules Mining
  - Collaborative Filtering

Supervised Learning

- Trying to predict unknown data attributes (outcomes) based on known attributes (predictors) for an entity
- Model built based on training data (past data) where outcomes and predictors are known
- Model used to predict future outcomes
- Types
  - Regression (continuous outcome values)
  - Classification (outcome classes)

Supervised Learning Process



Training and Testing Data

- Historical Data contains both predictors and outcomes
- Split as training and testing data
- Training data is used to build the model
- Testing data is used to test the model
  - Apply model on testing data
  - Predict the outcome
  - Compare the outcome with the actual value
  - Measure accuracy
- Training and Test fit best practices
  - 70-30 split
  - Random selection of records. Should maintain data spread in both datasets

V2 Maestros
The Data Science Experts

Comparing Results

Confusion Matrix

- Plots the predictions against the actuals for the test data
- Helps understand the accuracy of the predictions
- Predictions can be Boolean or classes

		Actual		
		TRUE	FALSE	Total
Prediction	TRUE	44	6	50
	FALSE	9	41	50
	Total	53	47	100

Prediction Types

- The importance of prediction types vary by the domain
- True Positive (TP) and True Negative (TN) are the correct predictions
- False Negative (FN) can be critical in medical field
- False Positive (FP) can be critical in judicial field

		Actual	
		TRUE	FALSE
Prediction	TRUE	True Positive	False Positive
	FALSE	False Negative	True Negative

Confusion Matrix metrics

- Accuracy
  - Measures the accuracy of the prediction
  - Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Sensitivity
  - Hit rate or recall
  - Sensitivity = TP / (TP + FN)
- Specificity
  - True negative rate
  - Specificity = TN / (TN + FP)
- Precision
  - Precision = TP / (TP + FP)

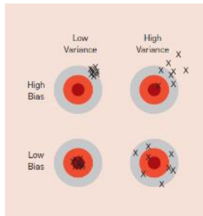
		Actual	
		TRUE	FALSE
Prediction	TRUE	True Positive	False Positive
	FALSE	False Negative	True Negative

V2 Maestros
The Data Science Experts

Prediction Errors

Bias and Variance

- Bias happens when the model "skews" itself to certain aspects of the predictors, while ignoring others. It is the error between prediction and actuals.
- Variance refers to the stability of a model – Keep predicting consistently for new data sets. It is the variance between predictions for different data sets.



Types of Errors

- In-Sample error is the prediction error when the model is used to predict on the training data set it is built upon.
- Out-of-sample error is the prediction error when the model is used to predict on a new data set.
- Over fitting refers to the situation where the model has very low in-sample error, but very high out-of-sample error. The model has "over fit" itself to the training data.

V2 Maestros
The Data Science Experts

Machine Learning with Spark

Overview

- Make ML practical and easy.
- Contains algorithms and utilities
- Packages
  - spark.mllib – original APIs built on RDDs
  - spark.ml – new higher level API built on DataFrames (Spark SQL) and pipelines
- http://spark.apache.org/docs/latest/ml-guide.html
- Special data types
  - Local vector
  - Labeled Point

Local Vector

- A vector of double values
- Dense Vector
  - (1.0, 3.0, 4.5)
- Sparse Vector
  - Original : (1.0,0.0,0.0,2.0,0.0)
  - Representation: (5, (0,3), (1.0,2.0)

Labeled Point

- Represents a Data point in ML
- Contains a "label" (the target variable) and a list of "features" (the predictors)
- LabeledPoint(1.0, Vectors.dense(1.0,0.0,3.0))

Pipelines

- A pipeline consist of a series of transformations and actions that need to be performed to create a model
  - DataFrame (the source)
  - Transformers (data transformations)
  - Estimators (model building)
  - Parameters (common parameters across algorithms)
- Internally optimized for better parallelism and resource utilization

Typical Spark ML workflow

- Load data into RDD
- Transform RDD
  - Filtering features
  - Strings to float
  - Indicator variables
  - Centering and Scaling
- Convert to LabeledPoint and create a DataFrame (label, features)
- Split training and testing
- Create model
- Perform predictions.

Linear Regression

Linear Relationships

Regression Analysis

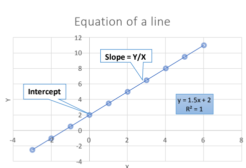
- Method of investigating functional relationship between variables
- Estimate the value of dependent variables from the values of independent variables using a relationship equation
- Used when the dependent and independent variables are continuous and have some correlation.
- Goodness of Fit analysis is important.

Linear Equation

- X is the independent variable
- Y is the dependent variable
- Compute Y from X using

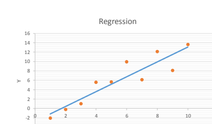
  $Y = \alpha X + \beta$

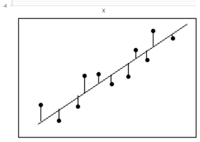
Coefficients:
- α = Slope = Y/X
- β = Intercept = value of Y when X=0



Fitting a line

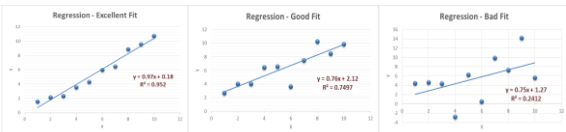
- Given a scatter plot of Y vs X, fit a straight line through the points so that the sum of square of vertical distances between the points and the line (called residuals) is minimized
- Best line = least residuals
- A line can always be fitted for any set of points
- The equation of the line becomes the predictor for Y





Goodness of Fit

- R-squared measures how close the data is to the fitted line
- R-squared varies from 0 to 1. The higher the value, the better the fit
- You can always fit a line. Use R-squared to see how good the fit is
- Higher correlation usually leads to better fit



Multiple regression

- When there are more than one independent variable that is used to predict the dependent variable.
- The equation $Y = \beta + \alpha_1 {}^*X_1 + \alpha_2 {}^*X_2 + \ldots + \alpha_p {}^*X_p$
- Same process used for prediction as a single independent variable
- Different predictors have different levels of impact on the dependent variable

Using Linear Regression for ML

- ML Technique to predict continuous data – supervised learning
- Predictors and outcomes provided as input
- Data analyzed (training) to come up with a linear equation
  - Coefficients
  - Intercept
  - R-squared
- Linear equation represents to model.
- Model used for prediction
- Typically fast for model building and prediction

Summary – Linear Regression

Advantages

- Fast
- Low cost
- Excellent for linear relationships
- Relatively accurate Continuous variables

Shortcomings

- Only numeric/ continuous variables
- Cannot model non-linear / fuzzy relationships
- Sensitive to outliers

Used in

- Oldest predictive model used in a wide variety of applications to predict continuous values

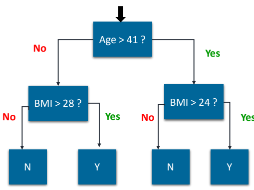
Decision Trees

Overview

- The simplest, easy to understand and easy to explain ML technique.
- Predictor variables are used to build a tree that would progressively predict the target variable
  - Trees start with a root node that start the decision making process
  - Branch nodes refine the decision process
  - Leaf nodes provide the decisions
- Training data is used to build a decision tree to predict the target
- The tree becomes the model that is used to predict on new data

Example

Age	BMI	Is Diabetic
24	22	N
33	28	N
41	36	Y
48	24	N
58	31	Y
61	35	Y



Predicting "Is Diabetic ?"

Choosing the right Predictors

- The depth of trees are highly influenced by the sequence in which the predictors are chosen for decisions
- Using predictors with high selectivity gives faster results
- ML implementations automatically make decisions on the sequence /preference of predictors

Summary – Decision Trees

Advantages
- Easy to interpret and explain
- Works with missing data
- Sensitive to local variations
- Fast

Shortcomings
- Limited Accuracy
- Bias builds up pretty quickly
- Not good with large predictors

Used in
- Credit approvals
- Situations with legal needs to explain decisions
- Preliminary categorization

V2 Maestros
The Data Science Experts

Naïve Bayes

Bayes' theorem (too) simplified

- Probability of an event A = P(A) is between 0 and 1
- Bayes' theorem gives the conditional probability of an event A given event B has already occurred.

  P(A/B) = P(A intersect B) * P (A) /P(B)
- Example
  - There are 100 patients
  - Probability of a patient having diabetes is P(A) = .2
  - Probability of patient having diabetes (A) given that the patient's age is > 50 (B) is P(A/B) = .4

Naïve Bayes Classification

- Application of Bayes' theorem to ML
- The target variable becomes event A
- The predictors become events B1 – Bn
- We try to find P(A / B1-Bn)

Age	BMI	Is Diabetic	
24	22	N	Probability of Is Diabetic = Y given that Age = 24 and BMI = 22
41	36	Y	Probability of Is Diabetic – Y given that Age = 41 and BMI = 36

Model building and prediction

- The model generated stores the conditional probability of the target for every possible value of the predictor.

	Overall	Age						Gender	
Salary		1 to 20	20 to 30	30 to 40	40 to 50	50 to 60	60 to 100	Female	Male
< 50K	.75	0.1	0.3	0.25	0.17	0.1	0.08	0.39	0.61
> 50K	.25	0.03	0.08	0.3	0.32	0.2	0.07	0.15	0.85
Overall		.08	.24	.26	.21	.12	.08	.33	.67

- When a new prediction needs to be done, the conditional probabilities are applied using Bayes' formula to find the probability
  - To predict for Age = 25
  - P(Salary < 50K / Age=25) = 0.3 * 0.75 / 0.24 = ~ 0.92
  - P(Salary > 50K / Age=25) = 0.08 * 0.25 / 0.24 = ~ 0.08

Summary – Naïve Bayes

Advantages
- Simple and fast
- Works well with noisy and missing data
- Provides probabilities of the result
- Very good with categorical data

Shortcomings
- Limited Accuracy
- Expects predictors to be independent
- Not good with large numeric features

Used in
- Medical diagnosis
- Spam filtering
- Document classification
- Sports predictions

V2 Maestros
The Data Science Experts

Random Forests

Overview

- Random Forest is one of the most popular and accurate algorithms
- It is an Ensemble method based on decision trees
  - Builds multiple models – each model a decision tree
  - For prediction – each tree is used to predict an individual result
  - A vote is taken on all the results to find the best answer

How it works

- Lets say the dataset contains m samples (rows) and n predictors (columns)
- x trees are built, each with a subset of data
- For each tree, a subset of m rows and n columns are chosen randomly.
- For example, if the data has 1000 rows and 5 columns, each tree is built using 700 rows and 3 columns
- The data subset is used to build a tree
- For prediction, new data is passed to each of the x trees and x possible results obtained
- For example, if we are predicting buy=Y/N and there are 500 trees, we might get 350 Y and 150 N results
- The most found result is the aggregate prediction.

Summary – Random Forest

Advantages
- Highly accurate
- Efficient on large number of predictors
- Fully parallelizable
- Very good with missing data

Shortcomings
- Time and Resource consuming
- For categorical variables, bias might exist if levels are disproportionate

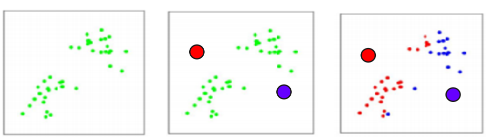
Used in
- Scientific Research
- Competitions
- Medical Diagnosis

K-means Clustering

Overview

- Unsupervised Learning technique
- Popular method for grouping data into subsets based on the similarity
- Partitions n observations with m variables into k clusters where by each observation belongs to only one cluster
- How it works
  - An m dimensional space is created
  - Each observation is plotted based on this space based on the variable values
  - Clustering is done by measuring the distance between points and grouping them
- Multiple types of distance measures available like Euclidian distance and Manhattan distance

Clustering - Stages

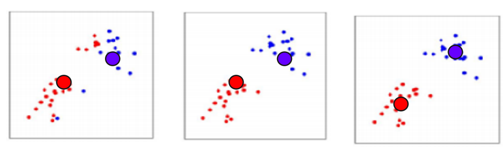


- Dataset contains only m=2 variables. We will create k=2 clusters
- Plot observations on a two dimensional plot

- Choose k=2 centroids at random
- Measure the distance between each observation to each centroid

- Assign each observation to the nearest centroid
- This forms the clusters for round 1

Clustering - Stages



- Find the centroid of each of the cluster
- Centroid is the point where the sum of distances between the centroid and each point is minimum

- Repeat the process of finding the distance between each observation to each centroid (the new one) and reassign each point to the nearest one

- Find the centroid for the new clusters
- Repeat the process until the centroids don't move

Summary – K-means clustering

Advantages
- Fast
- Efficient with large number of variables
- Explainable

Shortcomings
- K needs to be known
- The initial centroid position has influence on clusters formed

Used in
- Preliminary grouping of data before other classification
- General grouping
- Geographical clustering

Collaborative Filtering

V2 Maestros
The Data Science Experts

Association Rules Mining

V2 Maestros
The Data Science Experts

Overview

- ARM shows how frequently sets of items occur together
  - Find Items frequently brought together
  - Find fraudulent transactions.
  - Frequent Pattern Mining/ Exploratory Data Analysis
  - Finding the next word
- One of the clustering techniques
- Assumes all data are categorical, not applicable for numeric data
- Helps generate association rules that can be then used for business purposes like stocking aisles.

V2 Maestros
The Data Science Experts

Datasets

- Market basket transactions
  - Tran 1 { bread, cheese, milk}
  - Tran 2 { apple, eggs, yogurt}
  - Tran 3 {bread, eggs}
- Text document data set (bag of words)
  - Doc 1 { cricket, sachin, India }
  - Doc 2 { soccer, messi, Barcelona}
  - Doc 3 { sachin, messi, superstars}

V2 Maestros
The Data Science Experts

ARM measures

- Let N be the number of transactions
- Let X, Y and Z be individual items
- Support measures how frequently an combination of items occurs in the transactions
  - Support(X) = count(transactions with X)/ N
  - Support(X,Y)= count(transactions with X and Y)/N
- Confidence measures the expected probability that Y would occur when X occurs
  - Confidence(X -> Y) = support(X,Y) / support(X)
- Lift measures how many more times X and Y occurs together than expected
  - Lift(X -> Y) = confidence(X->Y) / support(Y)

V2 Maestros
The Data Science Experts

Rules and goals

- A rule specifies when one item occurs the other too occurs
  - When bread is brought, milk is brought 33% of the time.
  - When India occurs in the bag of words, sachin occurs 20% of the time.
- Goal is to find all rules that satisfy the user specified minimum support and minimum confidence
- A frequent itemset is an itemset whose support is > the minimum support level specified.
- Apriori algorithm is the most popular ARM algorithm

V2 Maestros
The Data Science Experts

Data Formats

- Transaction form
  - a, b, c
  - a, c, d, e
  - a, d
- Table form

Attr1,	Attr2,	Attr 3
A	B	C
A	C	D

- Table should be converted to transaction

  (Attr1 = A), (Attr2 = B), (Attr3 = C)
  (Attr1 = A), (Attr2 = C), (Attr3 = D)

Dimensionality Reduction

Principal Component Analysis

Issues with too many predictors

- Memory requirements
- CPU requirements / time taken for machine learning algorithms
- Correlation between predictors
- Over fitting
- Some ML algorithms don't work fine with too many predictors

Manual selection

- Using domain knowledge
  - Purely based on hypothesis
  - Risky – there could be unknown correlations
- Using Correlation co-efficients
  - Variables with good correlation can only be picked up.
- Using Decision Trees
  - Decision trees are fast and choose variables based on correlation
  - Variables used in the decision trees can be picked for further processing

Principal Component Analysis

- Used to reduce the number of predictors
- Based on Eigen Vectors and Eigen Values.
- Given a set of M predictors, PCA transforms this to a set of N predictors such that N < M
- The new predictors are derived predictors called PC1, PC2, PC3
- The new predictors retain similar levels of correlation and predictability like the original predictors

Big Data Analytics with Apache Spark and Python

Closing Remarks

Course Structure

- Hadoop and Spark Concepts
- Spark Programming including Spark SQL and Spark Streaming
- Basics of Real Time Data Science
- Machine Learning with Spark
- End-to-End use cases
- Resource Bundle

V2 Maestros
The Data Science Experts

Next Steps

- Continue to learn on Data Science
  - Big Data
- Try exercises with new data sets
  - UCI Data sets
  - Kaggle
  - Crowd analytics
- Interviews
  - Focus on process

Congratulations on finishing this course !

We hope this course helps you to advance your career.

Best of luck !