Apache Spark

Session 12 - Machine Learning Library (MLlib) & GraphX

WELCOME - KNOWBIGDATA

- ❑ Expert Instructors
- ❑ CloudLabs
- ❑ Lifetime access to LMS
  - ❑ Presentations
  - ❑ Class Recording
  - ❑ Assignments + Quizzes
  - ❑ Project Work

- ❑ Real Life Project
- ❑ Course Completion Certificate
- ❑ 24x7 support
- ❑ KnowBigData - Alumni
  - ❑ Jobs
  - ❑ Stay Abreast (Updated Content, Complimentary Sessions)
  - ❑ Stay Connected

COURSE CONTENT

1	Big Data & Use Cases
2	Big Data Architectures + HDFS + YARN
3	MapReduce + NoSQL
4	HBASE + MongoDB
5	Apache Spark + Flume + Sqoop
6	Statistics & Statistical Inference
7	Understanding simple Analytics with SQL + BIRT
8	Analysing & Visualisation Data with R
9	Analysing & Visualisation Data with R (Adv)
10	Analyzing Big Data with Hive, Pig, Mahout
11	Analyzing Big Data with MLLib & SparkR
12	Advanced Visualization of Data with Tableau & D3JS

About Instructor?

2014	**KnowBigData**	Founded
2014 2012	**Amazon**	Built High Throughput Systems for Amazon.com site using in-house NoSql.
2012	**InMobi**	Built Recommender that churns 200 TB
2011	**tBits Global**	Founded tBits Global Built an enterprise grade Document Management System
2006 2002	**D.E.Shaw**	Built the big data systems before the term was coined
2002	**IIT Roorkee**	Finished B.Tech.

MACHINE LEARNING

“Programming Computers to optimize a Performance using Example Data or Past Experience”

- Branch of Artificial Intelligence
- Design and Development of Algorithms
- Computers Evolve Behaviour based on Empirical Data

MACHINE LEARNING - TYPES

Supervised Learning

Using Labeled training data, to create a Classifier that can predict output for unseen inputs.

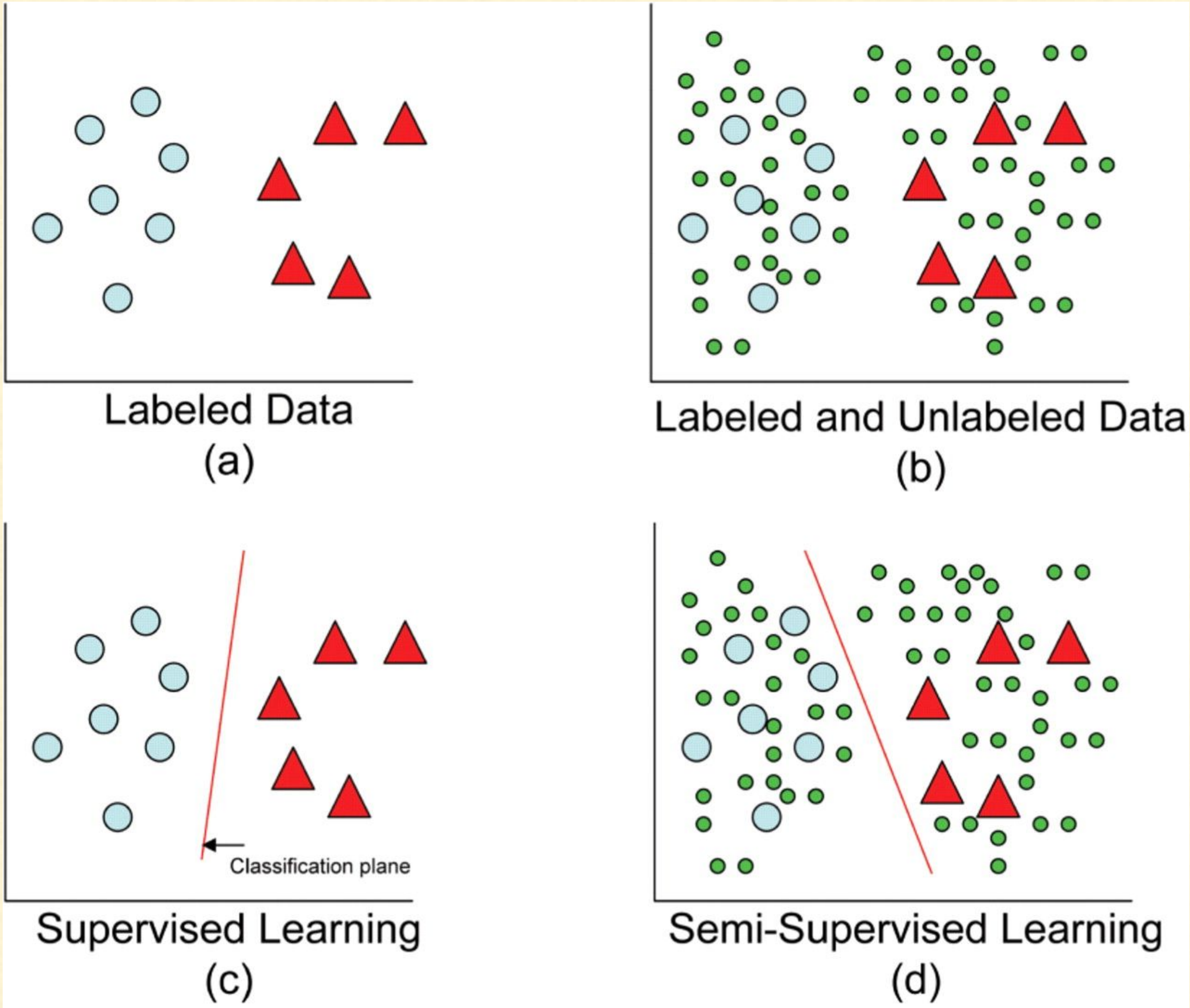
Unsupervised Learning

Using Unlabeled training data to create a function that can predict output.

Semi-Supervised Learning

Make use of unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data.

MACHINE LEARNING - TYPES



MACHINE LEARNING - APPLICATIONS

- Recommend Friends, Dates, Products to end-user.
- Classify content into pre-defined groups.
- Find Similar content based on Object Properties.
- Identify key topics in large Collections of Text.
- Detect Anomalies within given data.
- Ranking Search Results with User Feedback Learning.
- Classifying DNA sequences.
- Sentiment Analysis/ Opinion Mining
- Computer Vision.
- Natural Language Processing,
- BioInformatics.
- Speech and HandWriting Recognition.

MACHINE LEARNING - TOOLS

DATA SIZE	CLASSFICATION	TOOLS
Lines Sample Data	Analysis and Visualization	Whiteboard,…
KBs - low MBs Prototype Data	Analysis and Visualization	Matlab, Octave, R, Processing,
MBs - low GBs Online Data	Analysis	NumPy, SciPy, Weka, BLAS/LAPACK
	Visualization	Flare, AmCharts, Raphael, Protovis
GBs - TBs - PBs Big Data	Analysis	Mahout, Giraph MLlib, SparkR, GraphX

Machine Learning Library (MLlib)

Goal is to make practical machine learning scalable and easy

Consists of common learning algorithms and utilities, including:

- Classification
- Regression
- Clustering
- Collaborative filtering
- Dimensionality reduction
- Lower-level optimization primitives
- Higher-level pipeline APIs

Divided into two packages

spark.mllib

- *contains the original API built on top of RDDs.*

spark.ml

- *provides higher-level API built on top of DataFrames for constructing ML pipelines.*

spark.mllib - DataTypes

Local vector

integer-typed and 0-based indices and double-typed values

dv2 = [1.0, 0.0, 3.0]

Labeled point

a local vector, either dense or sparse, associated with a label/response

pos = LabeledPoint(1.0, [1.0, 0.0, 3.0])

Matrices:

Local matrix

Distributed matrix

RowMatrix

IndexedRowMatrix

CoordinateMatrix

BlockMatrix

spark.mllib - Basic Statistics

Summary statistics
Correlations
Stratified sampling
Hypothesis testing
Random data generation
Kernel density estimation
See https://spark.apache.org/docs/latest/mllib-statistics.html

spark.mllib - Basic Statistics - Summary

```
from pyspark.mllib.stat import Statistics
sc = ... # SparkContext

mat = ... # an RDD of Vectors

# Compute column summary statistics.
summary = Statistics.colStats(mat)
print(summary.mean())
print(summary.variance())
print(summary.numNonzeros())
```

MLlib - Classification and Regression

MLlib supports various methods:

Binary Classification

linear SVMs, logistic regression, decision trees, random forests, gradient-boosted trees, naive Bayes

Multiclass Classification

logistic regression, decision trees, random forests, naive Bayes

Regression

linear least squares, Lasso, ridge regression, decision trees, random forests, gradient-boosted trees, isotonic regression

[More Details>>]

MLlib - Collaborative Filtering

- Commonly used for recommender systems
- Techniques aim to fill in the missing entries of a user-item association matrix
- Supports model-based collaborative filtering,
- Users and products are described by a small set of latent factors
  - that can be used to predict missing entries.
- MLlib uses the alternating least squares (ALS) algorithm to learn these latent factors.

MLlib - Collaborative Filtering - Example

```
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel,
Rating
# Load and parse the data
data = sc.textFile("/data/spark/mllib/als/test.data")
ratings = data.map(lambda l: l.split(',')).map(lambda l: Rating(int(l[0]), int(l[1]),
float(l[2])))

# Build the recommendation model using Alternating Least Squares
rank = 10
numIterations = 10
model = ALS.train(ratings, rank, numIterations)
```

MLlib - Collaborative Filtering - Example

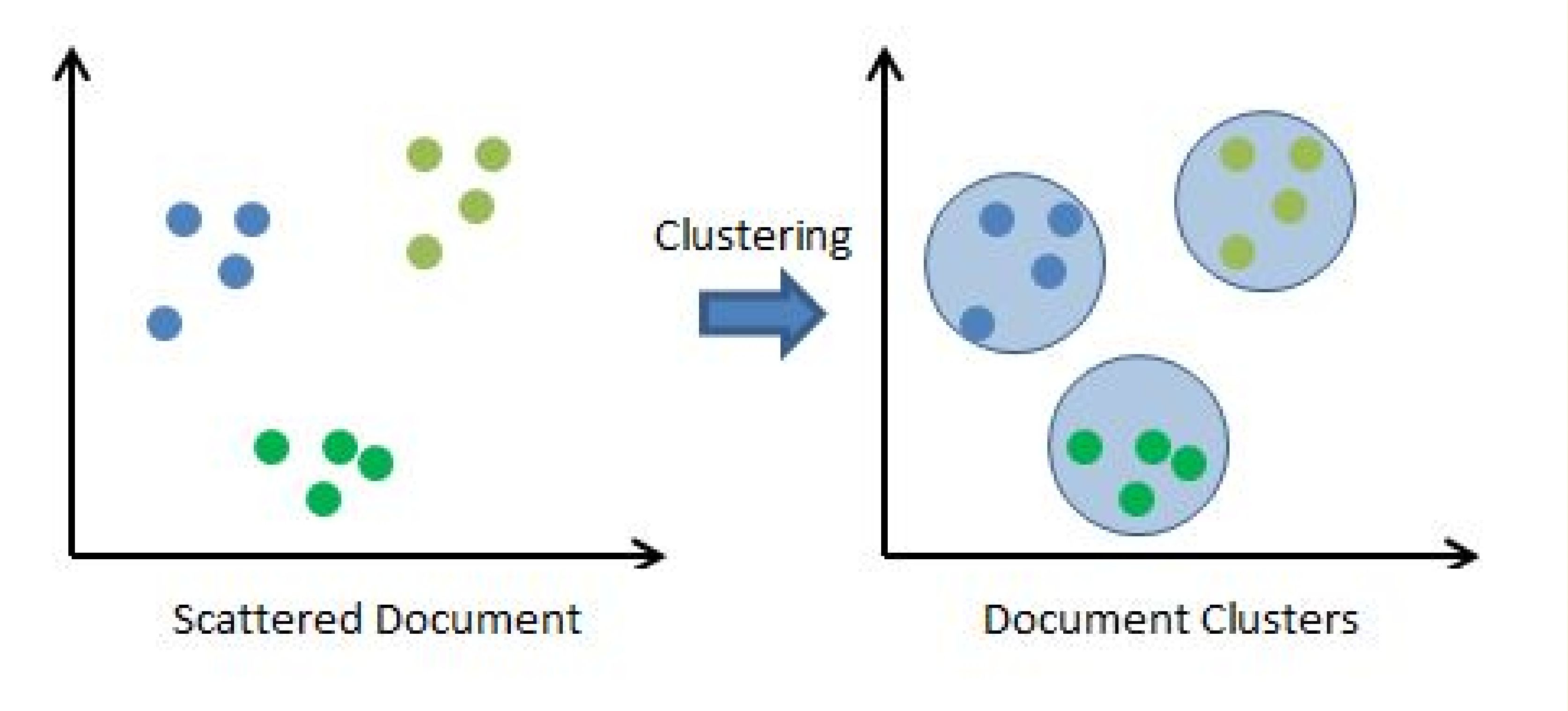
```
# Evaluate the model on training data
testdata = ratings.map(lambda p: (p[0], p[1]))
predictions = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
ratesAndPreds = ratings.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
print("Mean Squared Error = " + str(MSE))

# Save and load model
model.save(sc, "myModelPath")
sameModel = MatrixFactorizationModel.load(sc, "myModelPath")
```

See More

MLlib - Clustering

- Clustering is an unsupervised learning problem
- Group subsets of entities with one another based on some notion of similarity.
- Often used for exploratory analysis



MLlib supports the following models:

K-means

Clusters the data points into a predefined number of clusters

Gaussian mixture

Subgroups within overall population

Power iteration clustering (PIC)

Clustering vertices of a graph given pairwise similarities as edge properties

Latent Dirichlet allocation (LDA)

Infers topics from a collection of text documents

Streaming k-means

MLlib - k-means Example

```
from pyspark.mllib.clustering import KMeans, KMeansModel
from numpy import array
from math import sqrt

# Load and parse the data
data = sc.textFile("/data/spark/mllib/kmeans_data.txt")
parsedData = data.map(lambda line: array([float(x) for x in line.split(' ')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations=10, runs=10,
initializationMode="random")
```

MLlib - k-means Example

```
# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = parsedData.map(lambda point: error(point)).reduce(lambda x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))

# Save and load model
clusters.save(sc, "myModelPath")
sameModel = KMeansModel.load(sc, "myModelPath")
```

MlLib - Other Classes of Algorithms

Dimensionality reduction:
https://spark.apache.org/docs/latest/mllib-dimensionality-reduction.html

Feature extraction and transformation:
https://spark.apache.org/docs/latest/mllib-feature-extraction.html

Frequent pattern mining:
https://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html

Evaluation metrics:
https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html

PMML model export:
https://spark.apache.org/docs/latest/mllib-pmml-model-export.html

Optimization (developer):
https://spark.apache.org/docs/latest/mllib-optimization.html

GraphX

- For computation on graphs
- Extends the Spark RDD by introducing a new Graph abstraction
- Provides set of fundamental operators:
  - subgraph
  - joinVertices
  - aggregateMessages
- Has library of algorithms:
  - PageRank
    - If important pages link you, you are more important
  - Connected Components
    - Clusters amongst your facebook friends
  - Triangle Counting
    - Triangles passing through each vertex => measure of clustering.
- Only available in Scala yet

GraphX - Pagerank

```
// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "graphx/data/followers.txt")
// Run PageRank
val ranks = graph.pageRank(0.0001).vertices
// Join the ranks with the usernames
val users = sc.textFile("graphx/data/users.txt").map { line =>
  val fields = line.split(",")
  (fields(0).toLong, fields(1))
}
val ranksByUsername = users.join(ranks).map {
  case (id, (username, rank)) => (username, rank)
}
// Print the result
println(ranksByUsername.collect().mkString("\n"))
```

See more

Apache Spark

Thank you.

+1 419 665 3276 (US)
+91 803 959 1464 (IN)

reachus@knowbigdata.com

Subscribe to our Youtube channel for latest videos - https://www.youtube.com/channel/UCxugRFe5wETYA7nMH6VGyEA