BIG DATA

UNIT - 5

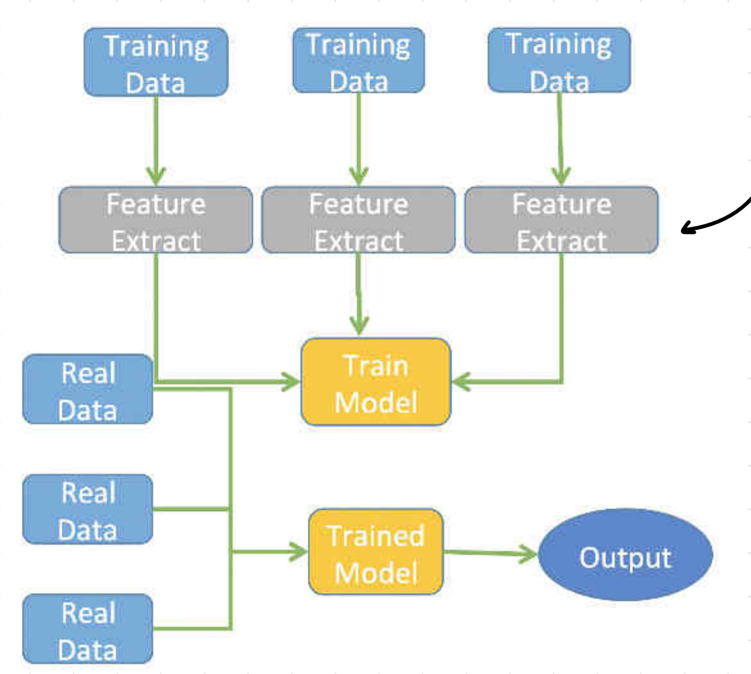
Advanced Analytics on Big Data

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

Machine Learning

- Complex programs



speech rec:
frequency
of syllabus

Q: You want to write a program to separate out the rocks into different categories.
What will your approach be?

- Record colour, size, weight, thickness etc
- Cluster (unsupervised)

- Classify some training data into groups
- KNN/ neural nets (supervised)

Supervised vs Unsupervised Learning

- Eg: grouping IPL batsmen
  - manually (opening, attacking): supervised
  - automatically : unsupervised

Q: Classify as supervised or unsupervised

- In Google News, grouping together similar articles.
  unsupervised

- Determining if a particular credit card transaction is fraudulent
  supervised

- Analyzing an image to determine if a lump is cancerous
  supervised

- Recommending a product based on what the user buys
  unsupervised

- Market segmentation: dividing customers into various groups
  unsupervised - no pre-defined groups
  supervised - pre-defined groups

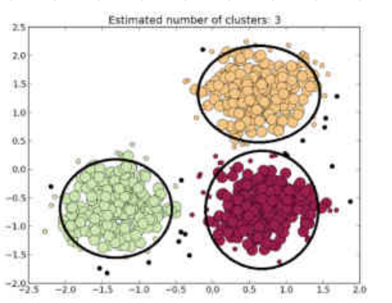
1. Scalable ML- K-means Clustering

- Partition dataset into k clusters

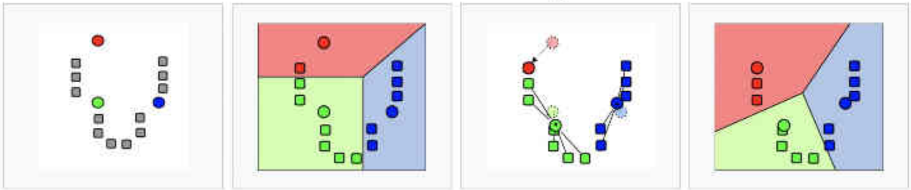
- Each training example is a point in d-dimensional space (d parameters / attributes)

- Use distance between points
  - Euclidean
  - Manhattan
  - Cosine

- Eg: 3-means



- Demonstration of k-means algorithm (iterative algorithm)



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Algorithm

1. Select k random centres

2. Assign each data point to its closest cluster centre – form k clusters

3. Re-compute cluster centroids (average of each dimension)

4. If new centres different from old centres, back to step 2

Q: • How can the k-means algorithm be modified to run with MapReduce?
• What is the output of Map and Reduce stages?
Hints:
• Iterative algorithm like page rank
• Which steps can be done in Map and which in Reduce?

Input: dataset – set of points – large
k centroids – small

Map: read both input files
assign points to centroids

output: <centroid, point>
key: centroid

Reduce: gets points associated with cluster
compute new centroids
output: <new_centroid>

Loop: compare old and new centroids
check max iterations

Q: Given the following points

- 20, 30, 99, 102,
- 53, 9, 11, 54

Partition them into two clusters using k-means assuming initial centroids are 20, 30.

- Assume that each row of numbers is on a different machine
- Show what the keys and values are for one iteration of k-means

Machine 1
points: 20, 30, 99, 102
clusters: 20, 30

Machine 2
points: 53, 9, 11, 54
clusters: 20, 30

Mapper 1 output
<1, 20>
<2, 30>
<2, 99>
<2, 102>

Mapper 2 output
<2, 53>
<1, 9>
<1, 11>
<2, 54>

Reducer input
<1, [20, 9, 11]>
<2, [30, 99, 102, 53, 54]>

Reducer output
<1, 13.33> OR 13.33
<2, 67.6> 67.6

K-means Optimisations

1. Use of Combiners

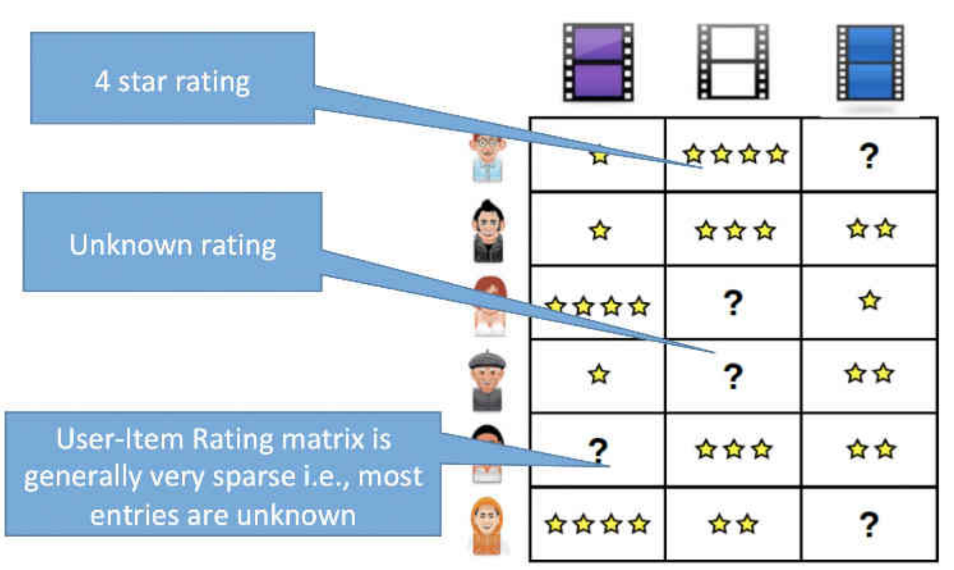
- compute local sums and count of the assigned clusters
- Sends <centroid, <partial_sum, count>>

2. Use of Single Reducer

- Amount of data to reducers small
- Single reducer to tell if clusters changed
- Single output file

2. Scalable ML - Alternating Least Squares

- Collaborative filtering
- User-item relationships (rating of each item by each vector)
- Sparse matrix



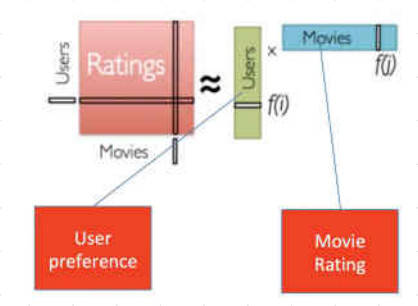
- User-Item rating matrix $R_{n\times m}$

- Try to write as a product of
  - User vector $A_{n\times 1}$
  - Item vector $B_{m\times 1}$

- Calculate A, B such that $R \approx AB$

- R : known
  A : unknown
  B : unknown

- r_{ij} : rating by user i for item j



- Approaches to factorisation
  - Gradient descent for optimisation problem (too slow)
  - Factorisation using alternating least squares

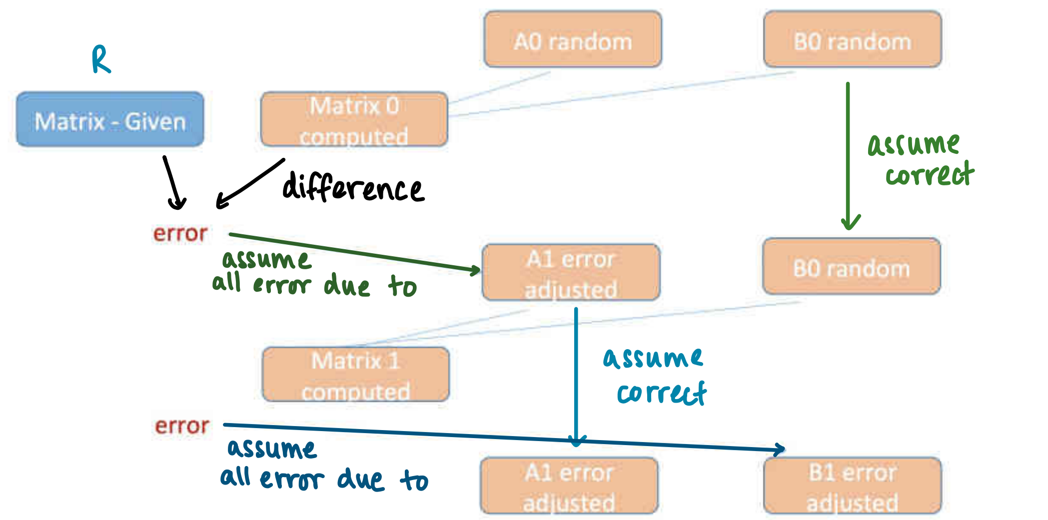
- Iterative algorithm with random initial assignments of A, B

- On i^{th} iteration, we have A_{i-1} and B_{i-1}
  - Assume B_{i-1} is correct and use to calculate best A_i
  - Assume A_i is correct and use to calculate best B_i

- Loop until convergence

ALS : ith Iteration

- Assume B_{i-1} correct
- Consider $R - A_i B_{i-1}^T$: error term
- $\| R - A_i B_{i-1}^T \|$ taken as $R - A_i B_{i-1}^T$ is a matrix and we want a number (absolute determinant)
- Find A_i that will minimise $\| R - A_i B_{i-1}^T \|$
- Shown that $A_i = (B_{i-1}^T B_{i-1})^{-1} B_{i-1}^T R^T$ (least squares regression estimate)
- Compute B_i from A_i



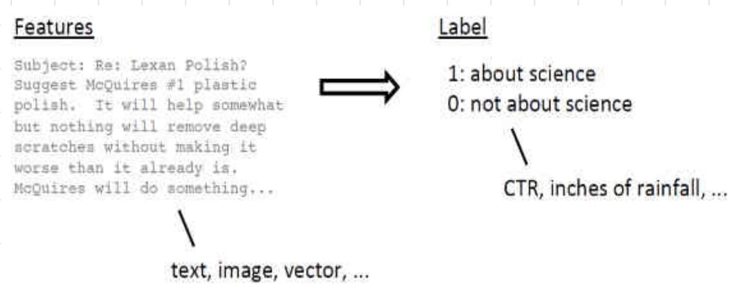
Q: ALS with Map-Reduce

- Use matrix multiplication with Map-Reduce to compute $A_i = (B_{i-1}^T B_{i-1})^{-1} B_{i-1}^T R^T$
- Similar for matrix inversion
- Compute-heavy

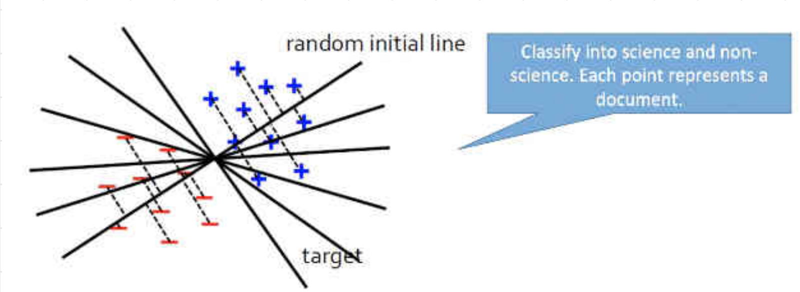
Spark MLLib

Text Classification

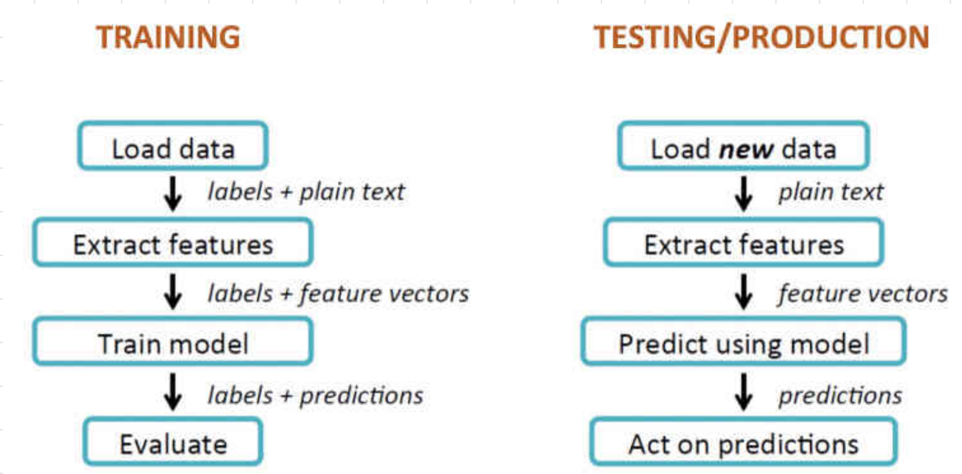
- Supervised - given document, predict its topic



- Logistic regression: find separation between 2 classes
  Support vector classifier: find line separating 2 classes



ML Workflow



Pain Points

- Handle many RDDs (each feature, labels)
- ML pipeline must be written as a script (not modular)
- Train many models on diff splits with diff hyperparameters

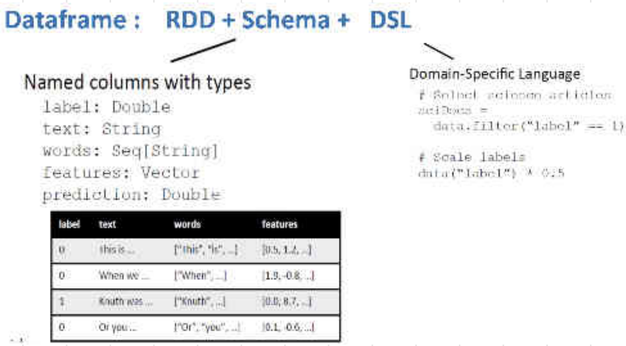
Solve Challenges

- Make RDDs easier to read
  - break up the fields
- Developers: program to extract features

1. Read RDDs into dataframes
2. ML pipeline: transformers, estimators, evaluators
3. Parameter tuning: API

Dataframes

- KBs to PBs
- Wide variety of formats (Hive, existing RDDs)
- SOTA optimisation and code generation via Spark SQL catalyst optimiser
- APIs in Python, Java
- RDD + Schema + DSL (domain specific language)



Spark ML Pipelines

- Automates process of defining models

ML Pipeline

1. Transformers
- Extract features from dataframe
- Features stored in new dataframe

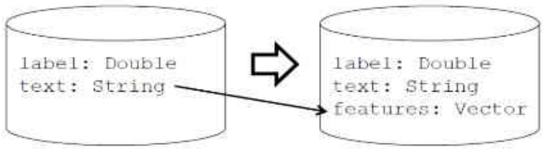
2. Estimators
- ML algorithms (standard defined or user defined)

3. Evaluators
- Compute predictions
- Estimate error metrics
- Tune algorithm parameters
- Evaluator depends on estimator (logistic regression, decision trees)

1. Transformers

- Extract features
- Input: dataframe, Output: dataframe

```
def transform(DataFrame): DataFrame
```



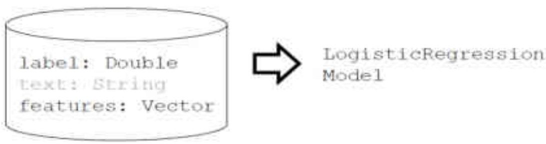
Label	Text
0	
1	

Label	Text	Features
0		
1		

2. Estimator

- Train model

```
def fit(DataFrame): Model
```

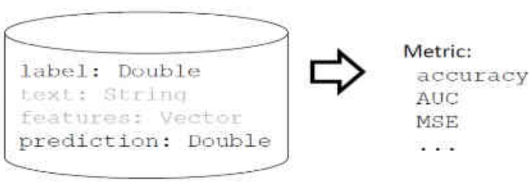


Label	Text	Features
0		
1		

3. Evaluator

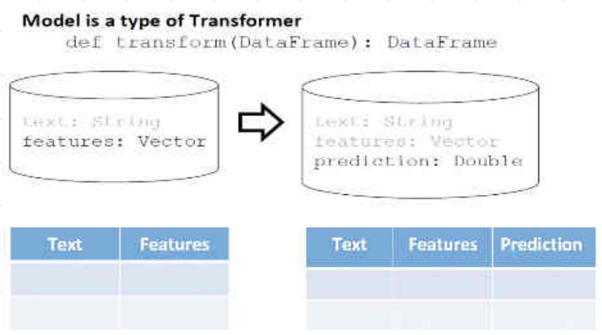
- Metrics

```
def evaluate(DataFrame): Double
```

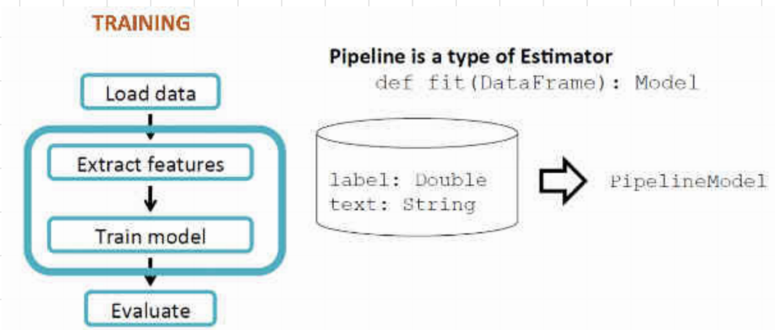


Label	Text	Features	Prediction
0			
1			

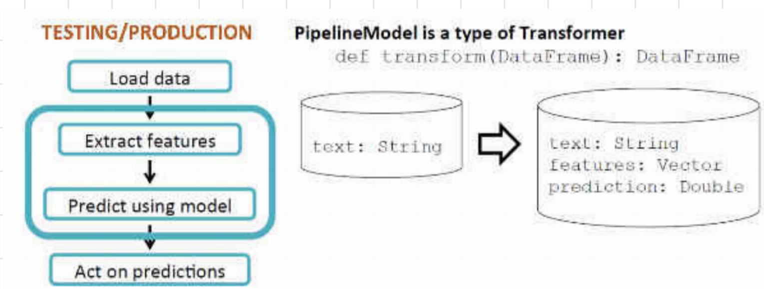
2. Predict Using Model (Test Phase)



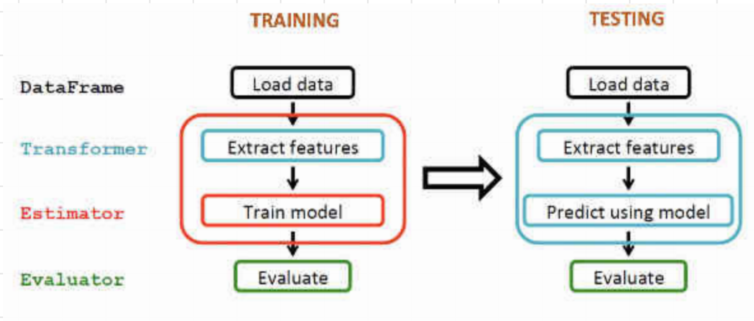
TRAINING PIPELINE



TESTING PIPELINE

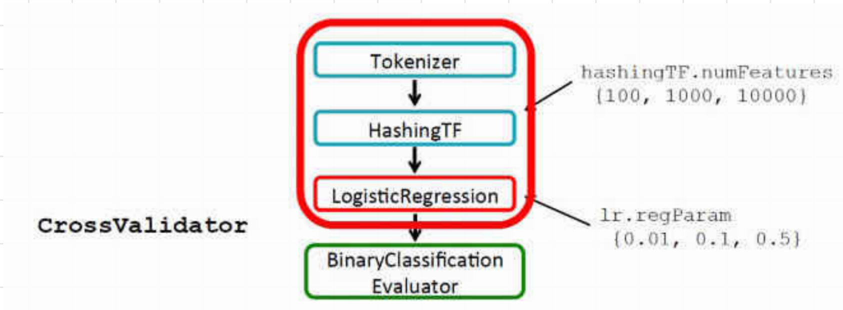


All Together



Parameter Tuning

- Given: estimator, parameter grid, evaluator
  - find best parameters



Q: Suppose we have a dataset in which each line has a recording of a noise, and its classification
- E.g., <bell.wav>, bell

What would be the input DataFrame be?

Suppose we want to recognize sounds by
- Extracting the frequencies from the wav file
- Gaussian model
  - Find the average frequency of each sound
  - For a new sound, calculate average frequency
  - Find closest matching sound

- What are the DataFrames, Evaluators, etc needed?

1. Input dataframe

<bell.wav, bell>

2. Feature dataframe

<bell.wav, bell, frequencies>

3. Transformer

<bell.wav, bell, avg-frequency>

4. Model
   - train (feature_dataframe): find avg freq for each sound type
   - predict (predict_dataframe): find closest matching sound

deep learning

- Iterative - matrix multiplication
- Challenges for DL
  - data on HDFS, train on Spark

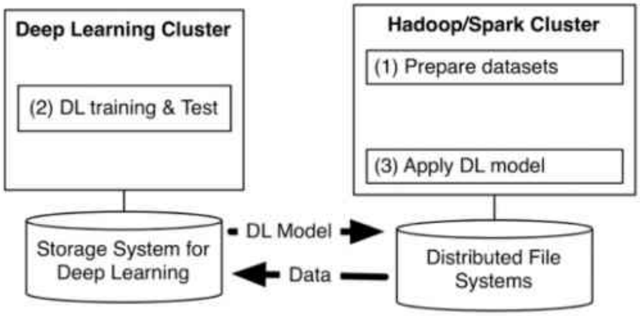


Figure 1: ML Pipeline with multiple programs on separated clusters

Tensorflow ON Spark

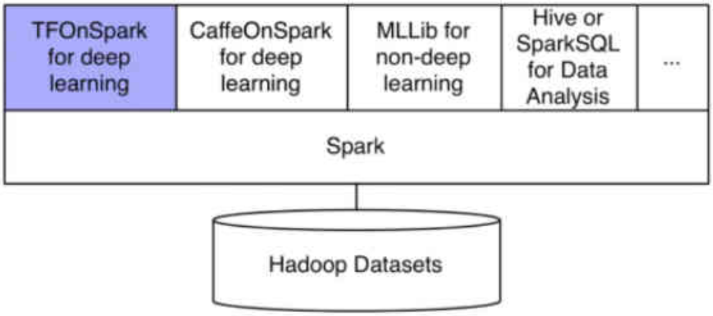


Figure 2: TensorFlowOnSpark for deep learning on Spark clusters

- Supports model & data parallelism
- TF runs on Spark executors
- Read data directly from HDFS
- RDMA: fast network transfers

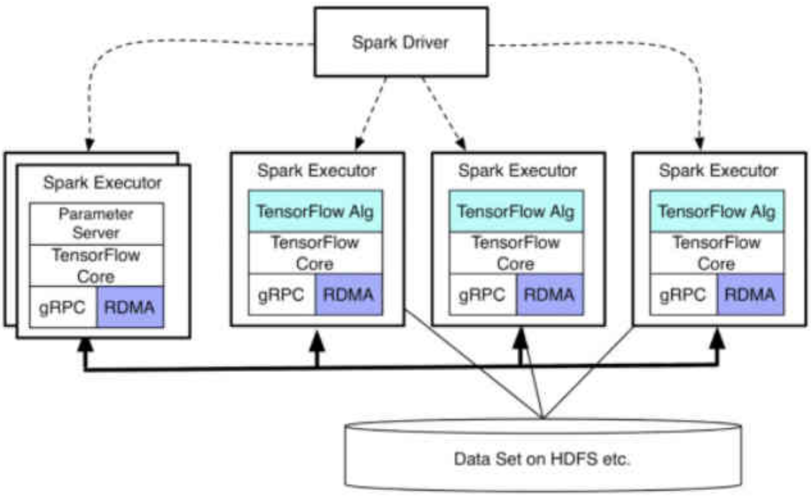


Figure 3: TensorFlowOnSpark system architecture