BIG DATA

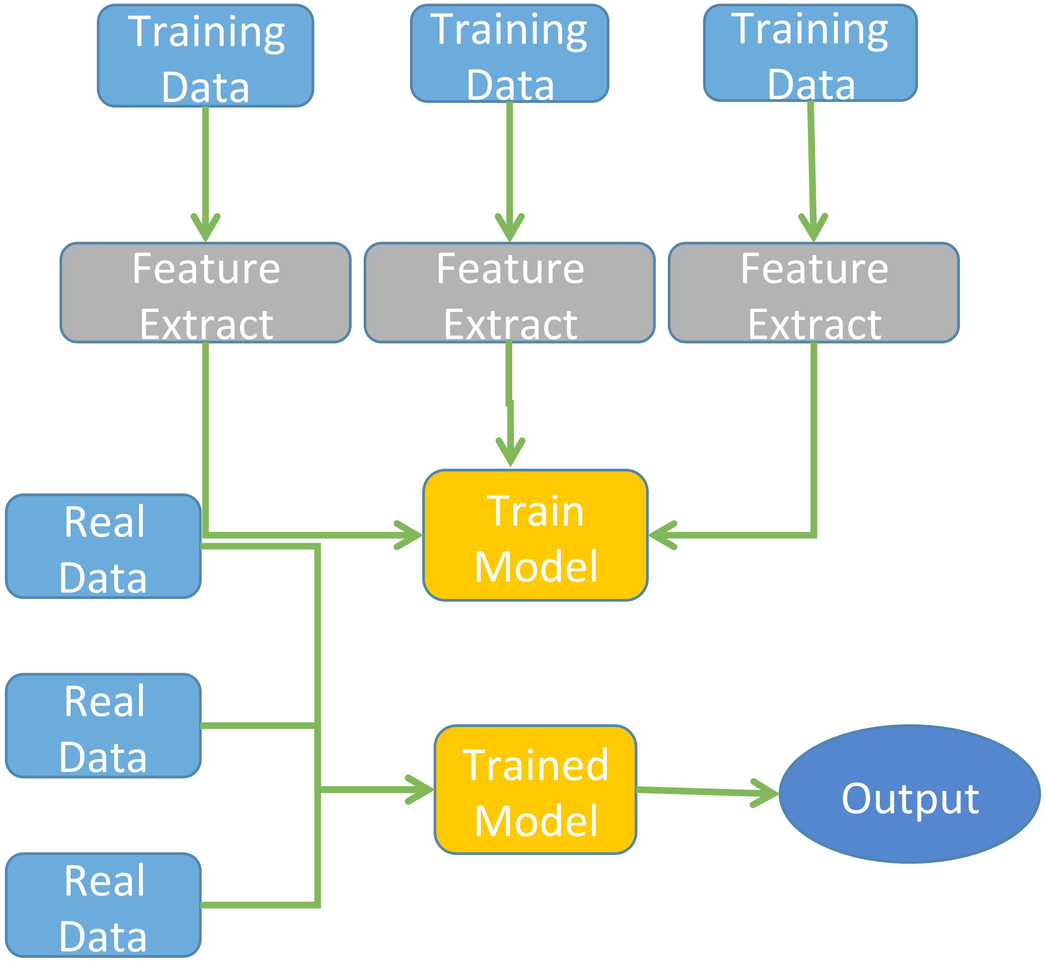
Machine Learning Algorithms At Scale - Clustering

K V Subramaniam

Computer Science and Engineering

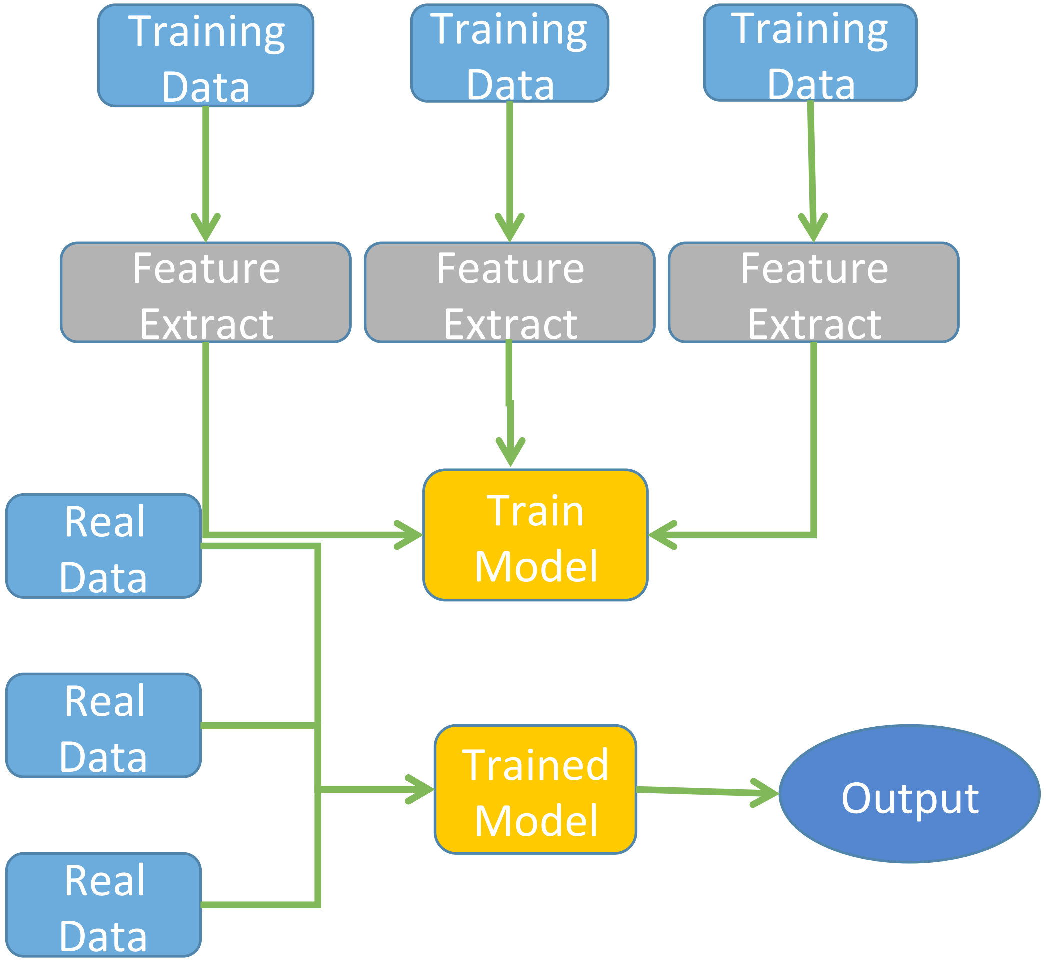
Why Machine Learning

- Sometimes, problems are complex
  - We don't want to write explicit programs
  - E.g., recognizing syllables in speech recognition
  - Theoretically, each syllable is a mixture of frequencies
  - Simpler to give the computer examples of syllables and ask the computer to "learn"
- Machine learning is an area of artificial intelligence



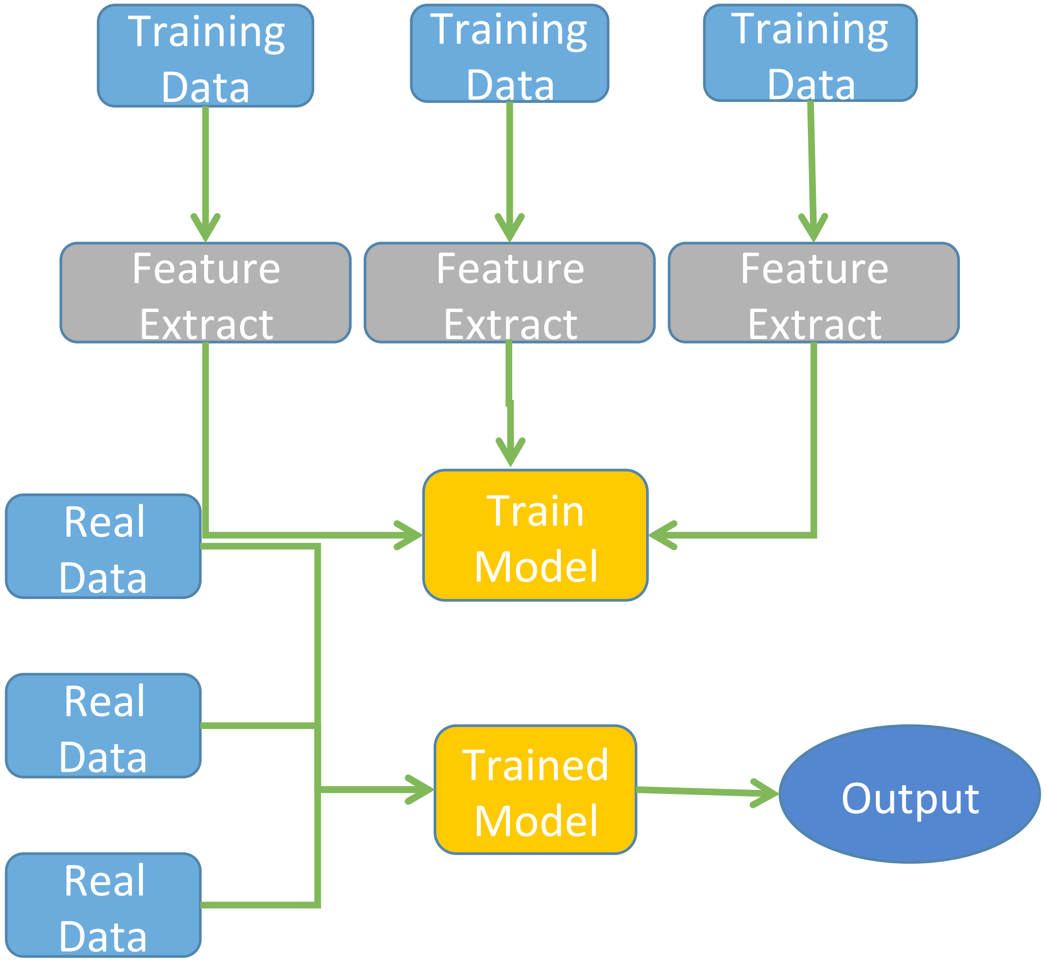
Overview of Machine Learning

- The examples we give are called *training data*
- The program may process the training data to calculate certain quantities called *features*
  - E.g., in speech recognition, the frequencies of each syllable or akshara
- It then uses the features to build a *model*
  - E.g., in speech recognition, which frequencies are associated with each syllable
  - Face recognition: for each person, e.g., how big are eyes, nose, mouth
- This process is called *training*



Overview of Machine Learning

- To see how good the model is, we can input test data to the program
  - E.g., input syllables to the model
- We can calculate the accuracy
  - How many syllables are correct
- If the accuracy is good, we can use the program in a product
- Input real data, get the output



Exercise 0

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

Types of Learning

- There are two types of learning
  - *Supervised* learning
  - *Unsupervised* learning

- We define the concepts we want the computer to learn
- Consider the photograph of pebbles on the right
  - We can input examples of each kind of pebble
    - Pebble 1 Large
    - Pebble 2 Medium
    - Pebble 3 Small
- The program will learn to classify pebbles

Unsupervised Learning

- The program
  - Computes various features of the pebbles
  - Groups similar pebbles together
  - It’s own classification of pebbles
- These may be different from the way a human being would classify them
- The same program can come up with different classifications if we change some parameters
  - E.g., if we ask the program to classify the pebbles into 3 groups or 4 groups
- Finds structure that’s already there in the data

Different Example required for this slide

- Why is unsupervised learning useful?
  - Recall the IPL class project
  - We asked you to group the batsmen into groups
  - We can manually define the groups; i.e., groups like “opener“ “attacking“ and so on
    - Supervised learning
  - Simpler to feed data about the batsmen and let the program group similar batsmen
    - Unsupervised learning

Supervised

- Input data is labeled
- Input training set consists of a pair
  - Data point or example
  - Classification

Unsupervised

- Input only training data points
- No labels
- Algorithm groups similar data points together

Review

- What is the basic method of machine learning and big data?
- Supervised vs Unsupervised learning

- What is the basic method of ML and big data?
  - Feature extraction
  - Model, train
  - Predict
- Supervised vs unsupervised learning
  - Predefined vs no predefined concepts

Exercise 1: 5 minutes

- Consider the list of machine learning applications on the right. Which use *supervised learning* and which use *unsupervised learning*?

- In Google News, grouping together similar articles.
- Determining if a particular credit card transaction is fraudulent
- Analyzing an image to determine if a lump is cancerous
- Recommending a product based on what the user buys
- Market segmentation: dividing customers into various groups

Exercise 1: Solution

- Consider the list of machine learning applications on the right. Which use *supervised learning* and which use *unsupervised learning*?

- 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. either depending on whether we already have pre-defined groups or not

Common ML Algorithms

Supervised

- Logistic regression
- Support Vector Machines
- Decision trees
- K-nearest neighbors

Unsupervised

- Principal Component Analysis
- Mixture models
- Hidden Markov models
- K-means

Scalable Machine Learning

- This class, focus on scalable or large-scale machine learning
  - Google search index (finding page rank over millions of pages)
  - Amazon recommendation (recommendations for millions of users over thousands of products)
- Challenges (as usual)
  - Data size is huge
  - Huge amount of computation
  - Failure is likely (huge amount of hardware)
- Solution
  - Use the right infrastructure (Hadoop, Spark,…)
  - Scalable algorithms
- In this class, we talk about *K-means* and *Alternating Least Squares* algorithm on MapReduce

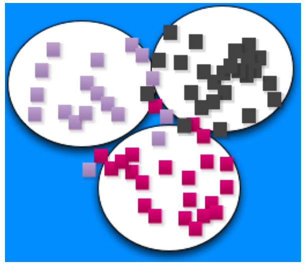
Scalable machine learning algorithms
- K-means introduction

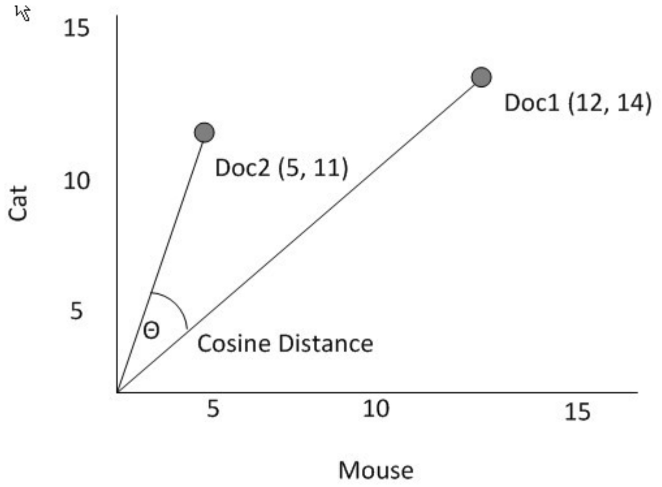
Clustering - Introduction

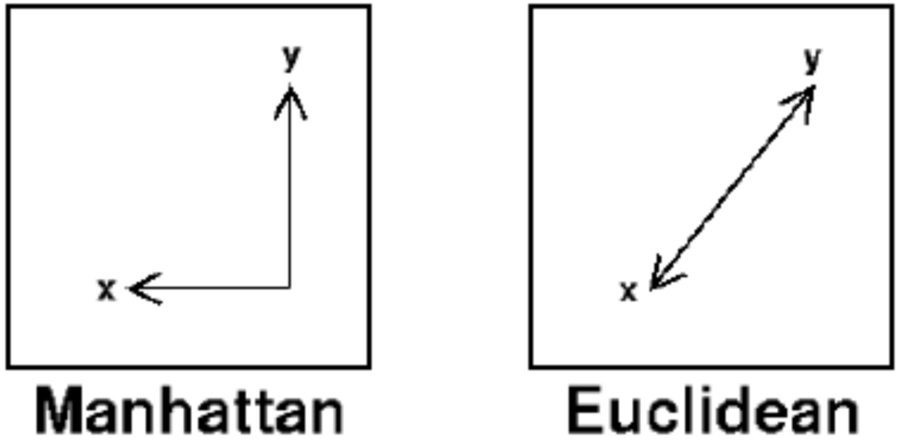
- Clustering
  - Partition a number of data points into related groups called clusters.
  - K-means clustering partitions a dataset into a specified number *k* of clusters
  - The points in a cluster should be similar to each other
- E.g., the IPL modeling, batsmen can be characterized by many parameters
  - E.g., strike rate, highest score, position (opener, …)
- To divide batsmen into groups, we need to be able to measure how similar batsmen are to each other

Distance Metrics

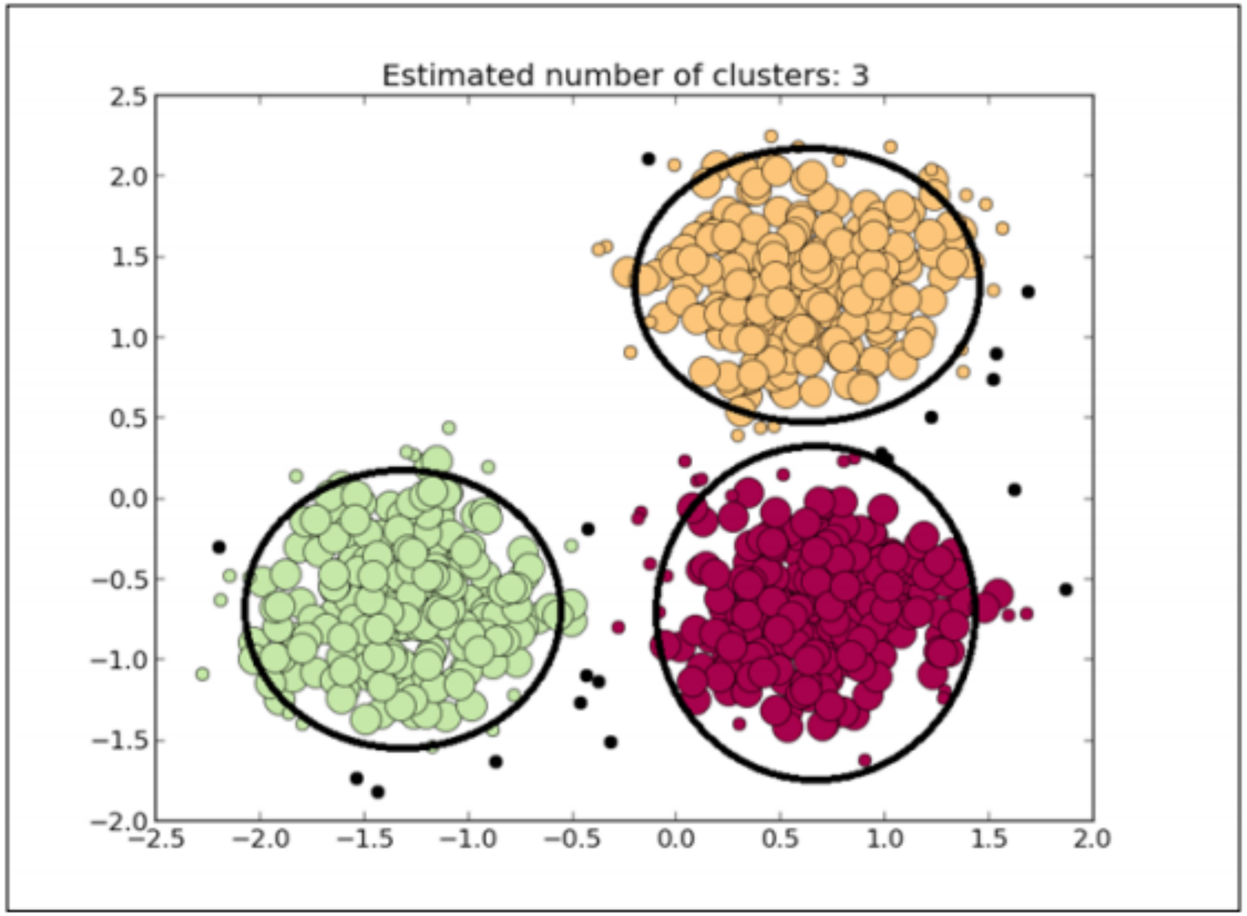
- We can consider each batsman to be a point in an *n*-dimensional space
  - *n* is the number of parameters we are measuring
- A *distance metric* measures the similarity (distance) between the two points
- For simplicity, consider a 2D space
  - Euclidean: Geometric distance
  - Manhattan: city blocks, used in traffic
  - Cosine: measures angle between points – used if points can have very different distances from origin







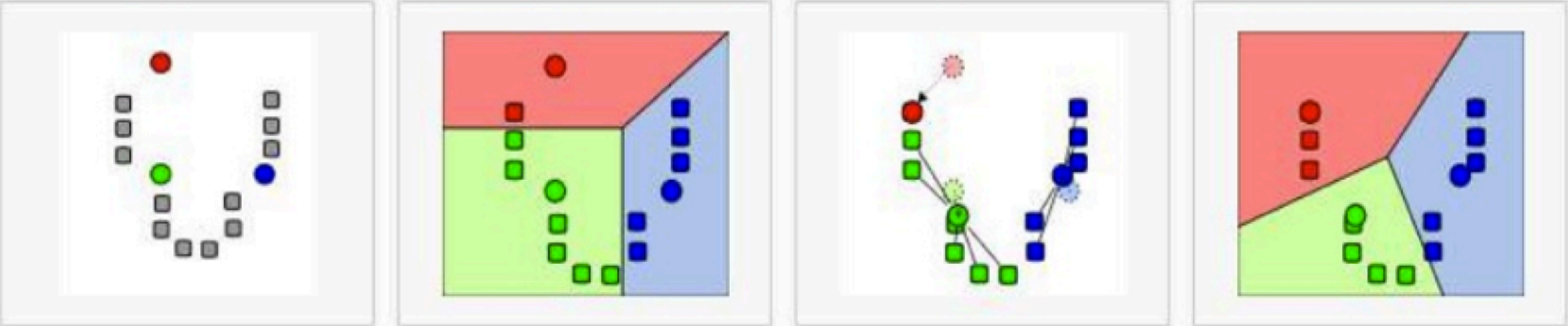
Example for clustering:



Note the outliers in the clusters.

K-Means Algorithm

Demonstration of the standard algorithm



1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (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 means.

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

Iterative algorithm until convergence

K-Means Algorithm

- **Initialize:** Select *K* points at random (Centers)
- **Step 1**: For each data point, assign it to the closest center
  - Now we formed *K* clusters
- **Step 2:** For each cluster, re-compute the centers
  - E.g., in the case of 2D points →
    - X: average over all x-axis points in the cluster
    - Y: average over all y-axis points in the cluster
- **Loop check:** If the new centers are different from the old centers (previous iteration) → Go to Step 1

Exercise 2: 10 minutes

- 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?

- **Initialize:** Select *K* points at random (Centers)
- **Step 1**: For each data point, assign it to the closest center
  - Now we formed *K* clusters
- **Step 2:** For each cluster, re-compute the centers
  - E.g., in the case of 2D points ➔
    - X: average over all x-axis points in the cluster
    - Y: average over all y-axis points in the cluster
- **Loop check:** If the new centers are different from the old centers (previous iteration) ➔ Go to Step 1

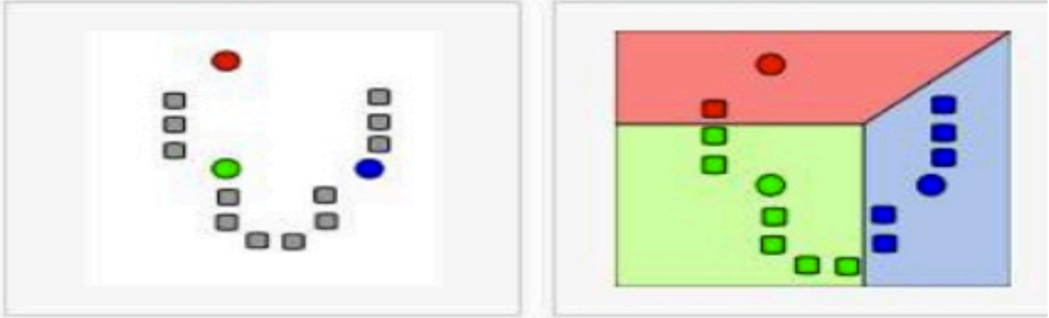
Scalable machine learning algorithms

- K-means with Map-Reduce

K-Means in MapReduce – 1/2

- **Input**
  - Dataset (set of points in 2D) --Large
  - Initial centroids (K points) --Small

- **Map (reads 2 files as input)**
  - Each map reads the K-centroids + one block from dataset
  - Assign each point to the closest centroid
  - Output <centroid, point> - centroid is the key

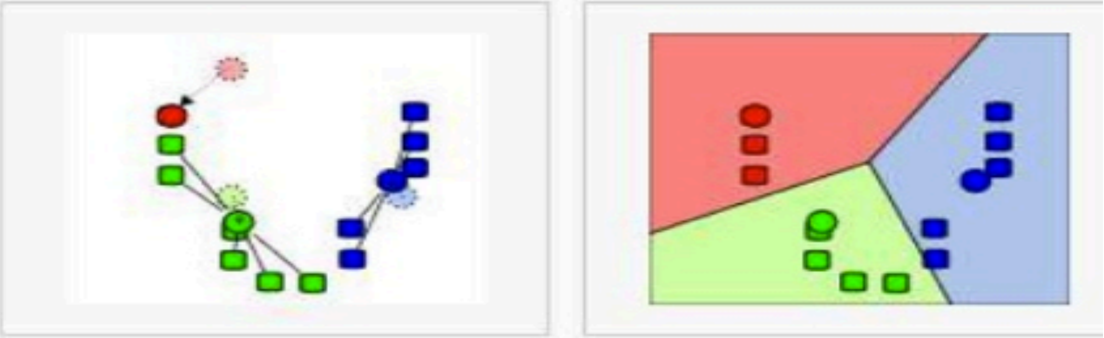


1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

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

K-Means in MapReduce 2/2

- **Reduce**
  - Gets all points for a given centroid
  - Re-compute a new centroid for this cluster
  - Output: <new centroid>
- **Loop check**
  - Compare the old and new set of K-centroids
    - If similar ➔ Stop
    - Else
      - If max iterations has reached ➔ Stop
      - Else ➔ Start another Map-Reduce Iteration



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

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

K-Means in MapReduce : Exercise

- 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

K-Means in MapReduce : Exercise

- Mapper1 output
  - 20, 20,
  - 30, 30,
  - 30, 99,
  - 30, 102
- Mapper 2 output
  - 30, 53,
  - 20, 9,
  - 20, 11,
  - 30, 54
- Reducer input
  - 20 , <20, 9, 11>
  - 30, <30, 99, 102, 53, 54>
- Reducer output
  - 13.33 and 67.6

Scalable machine learning algorithms
- K-means optimizations

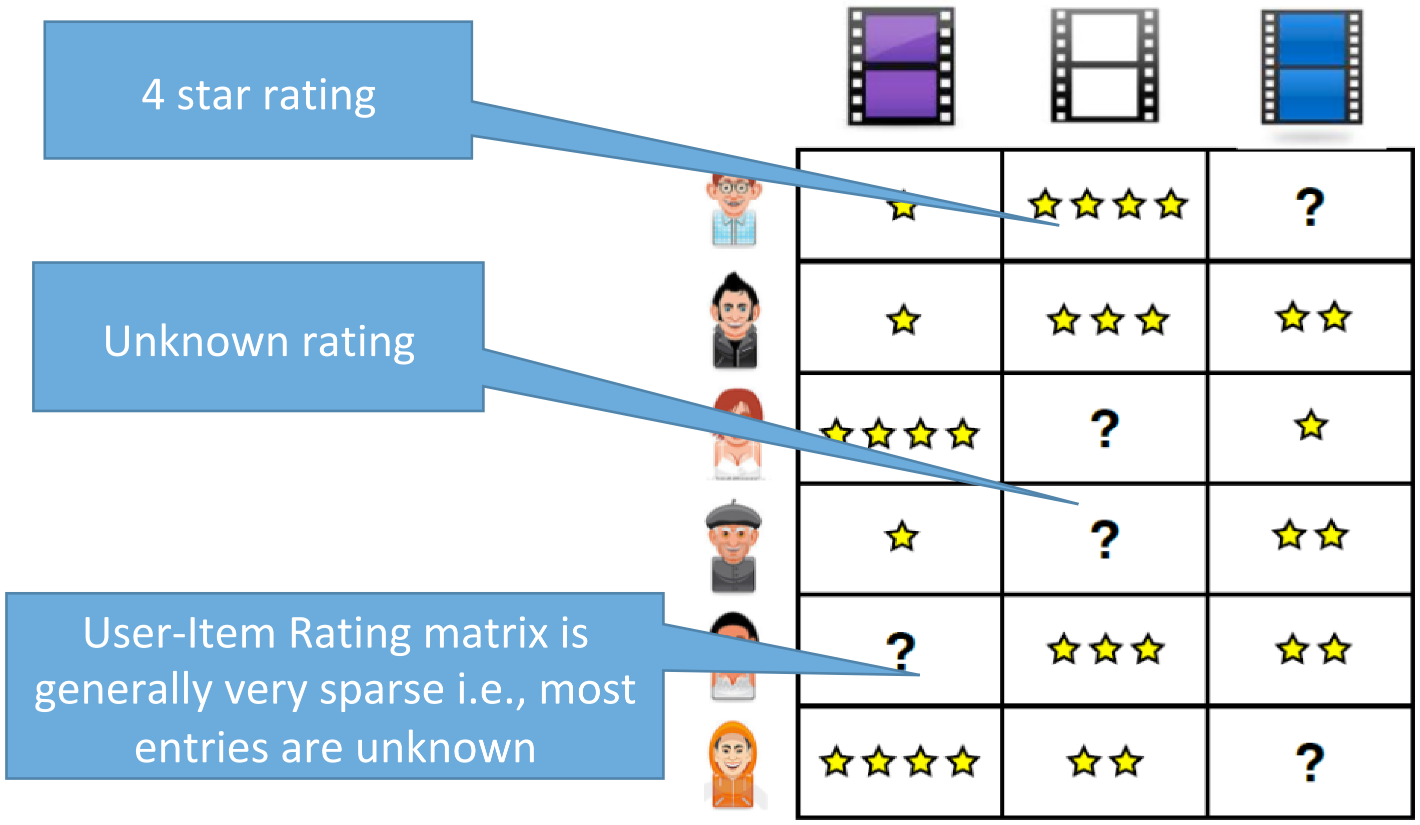
K-Means Optimizations

- **Use of Combiners**
  - Similar to the reducer
  - Computes for each centroid the local sums (and counts) of the assigned points
  - Sends to the reducer <centroid, <partial sums>>

- **Use of Single Reducer**
  - Amount of data to reducers is very small
  - Single reducer can tell whether any of the centers has changed or not
  - Creates a single output file

Scalable machine learning algorithms - Alternating least squares

Collaborative Filtering



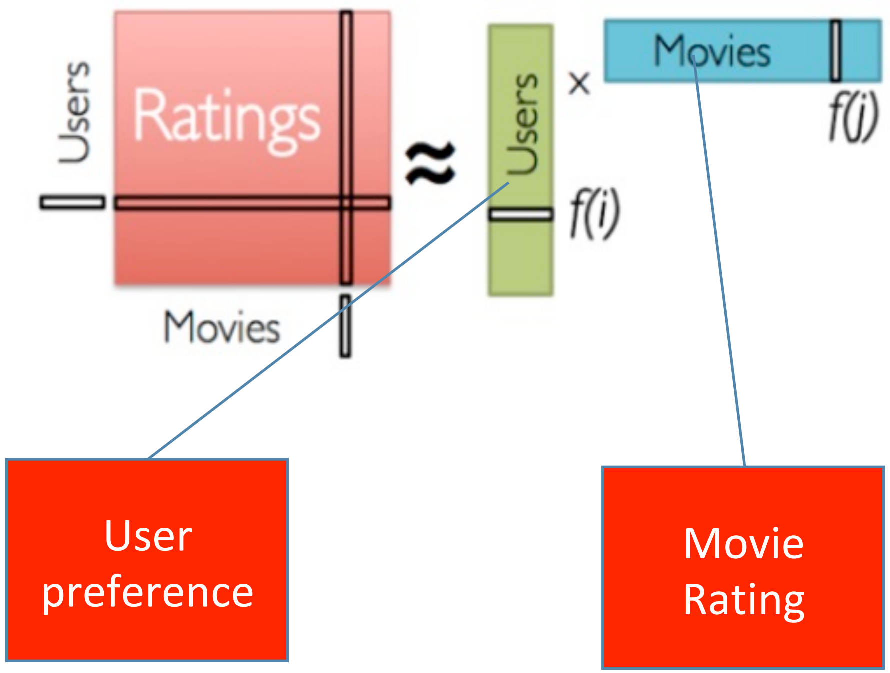
- Recover a rating matrix from a subset of its entries.

last.fm NETFLIX PANDORA Linkedin amazon.com

Xiangrui Meng, *MLLib: scalable machine learning on Spark*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Collaborative Filtering – ALS algorithm 1

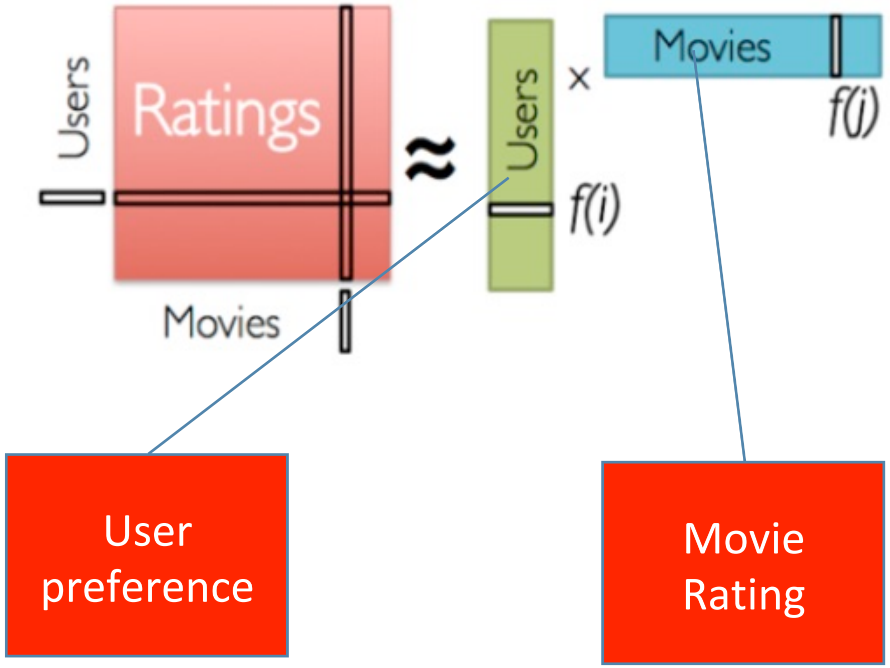
- Express User-Item Rating matrix (R) as a product of
  - User vector (A) dimension n=no of users
  - Item vector(B) dimension m=no of movies
  - Calculate A,B such that $R \approx AB$
  - A is a $nx1$ vector, B is a $1xm$ vector
  - R will be an nxm vector
- Suppose we need to find r_{ij} which is unknown
  - This is the rating of user i for item j
  - Calculate $R' = AB$
  - Use the ij^{th} element of R'



Xiangrui Meng, *MLLib: scalable machine learning on Spark*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Collaborative Filtering – Matrix Factorization

- Both A and B are not known
- This is like an optimization problem and can use Gradient Descent
    - But GD is too slow.
- Alternative – Factorize the matrix R into A and B
- We have to factorize R to get
    - A and B



Xiangrui Meng, *MLLib: scalable machine learning on Spark*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Alternating Least Squares – 1/2

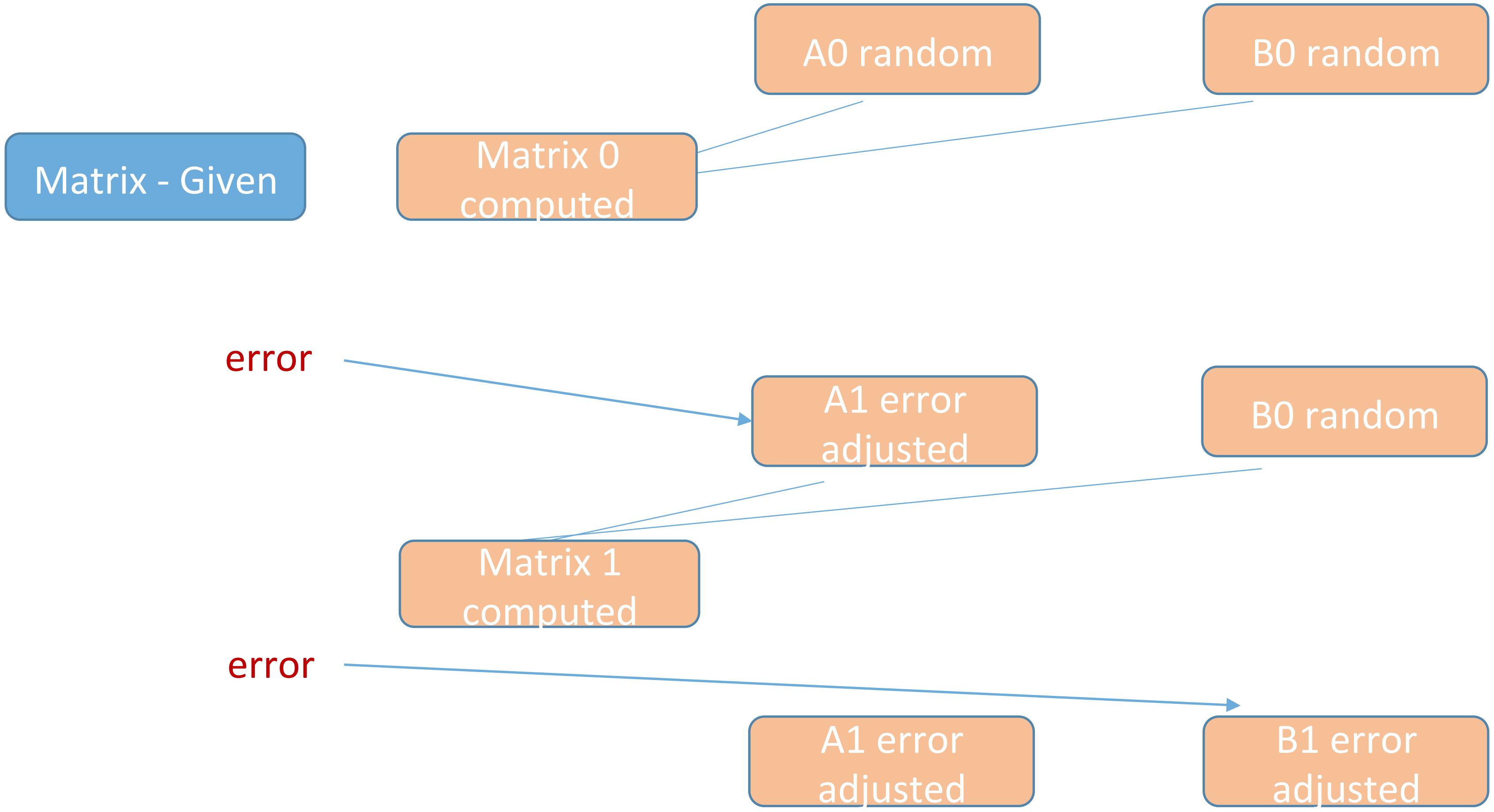
- Start with Random *A* and *B*
- The algorithm will loop until the correct value is calculated
  - Each iteration, the program will calculate new values for *A* and *B.*
  - Let A_i and B_i be the values of *A* and *B* on the i^{th} iteration of the loop.
- On the i^{th} iteration of the loop
  - We have calculated A_{i-1} and B_{i-1} on the previous iteration
  - Step 1: assume B_{i-1} is correct. Calculate best value for A_i
  - Step 2: assume A_i is correct. Calculate best value for B_i
  - Loop until converged

https://en.wikipedia.org/wiki/Ordinary_least_squares

Alternating Least Squares – 2/2

- On the i^{th} iteration of the loop
  - We have calculated A_{i-1} and B_{i-1} on the previous iteration
  - Step 1: assume B_{i-1} is correct. Calculate best value for A_i. How???
    - Consider $R - A_iB_{i-1}^T$
    - B_{i-1} and R are fixed. For any value of A_i, we can find $R - A_iB_{i-1}^T$
    - For any value of A_i, $R - A_iB_{i-1}^T$ is like an error term
      - The difference between R (the correct rating) and $A_iB_{i-1}^T$
    - The smaller the value of $R - A_iB_{i-1}^T$, the better
    - Since $R - A_iB_{i-1}^T$ can be –ve, we take $||R - A_iB_{i-1}^T||$ (determinant) and find A_i that will minimize
    - It can be shown that the solution is $A_i=(B_{i-1}^T B_{i-1})^{-1} B_{i-1}^T R^T$
    - Similarly for B_i
    - For the mathematics lovers, this is a least squares regression estimate

https://en.wikipedia.org/wiki/Ordinary_least_squares


A0 random
B0 random
Matrix - Given
Matrix 0 computed
error
A1 error adjusted
B0 random
Matrix 1 computed
error
A1 error adjusted
B1 error adjusted

Exercise 3: 10 minutes

- How can the ALS algorithm be modified to run with MapReduce?

- Start with Random A and B
- The algorithm will loop until the correct value is calculated
  - Each iteration, the program will calculate new values for A and B.
  - Let A_i and B_i be the values of A and B on the i^{th} iteration of the loop.
- On the i^{th} iteration of the loop
  - We have calculated A_{i-1} and B_{i-1} on the previous iteration
  - Step 1: assume B_{i-1} is correct. Calculate best value for $A_i = (B_{i-1}^T B_{i-1})^{-1} B_{i-1}^T R^T$
  - Step 2: assume A_i is correct. Similarly calculate best value for B_i
  - Loop until converged

Scalable machine learning algorithms - Alternating least squares with MR

Exercise 3:Solution

How can the ALS algorithm be modified to run with MapReduce?

Solution

- In Step 1, A_i is calculated by doing a number of matrix multiplications and inversions
- We have studied how to do matrix multiplication using MapReduce
- There are similar algorithms for doing matrix inverse using MapReduce

- Start with Random *A* and *B*
- The algorithm will loop until the correct value is calculated
  - Each iteration, the program will calculate new values for *A* and *B.*
  - Let A_i and B_i be the values of *A* and *B* on the i^{th} iteration of the loop.
- On the i^{th} iteration of the loop
  - We have calculated A_{i-1} and B_{i-1} on the previous iteration
  - Step 1: assume B_{i-1} is correct. Calculate best value for $A_i = (B_{i-1}^T B_{i-1})^{-1} B_{i-1}^T R^T$
  - Step 2: assume A_i is correct. Similarly calculate best value for B_i
  - Loop until converged

THANK YOU

K V Subramaniam, Usha Devi
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu
ushadevibg@pes.edu

BIG DATA

Machine learning Case Study

Spark MLLib

K V Subramaniam

Computer Science and Engineering

Motivational Problem: Text Classification

Goal : Given a text Document , Predict its topic.

Dataset: "20 Newsgroups"
From UCI KDD Archive

Features

```
Subject: Re: Lexan Polish?
Suggest McQuires #1 plastic
polish.  It will help somewhat
but nothing will remove deep
scratches without making it
worse than it already is.
McQuires will do something...
```

text, image, vector, ...

$\Longrightarrow$

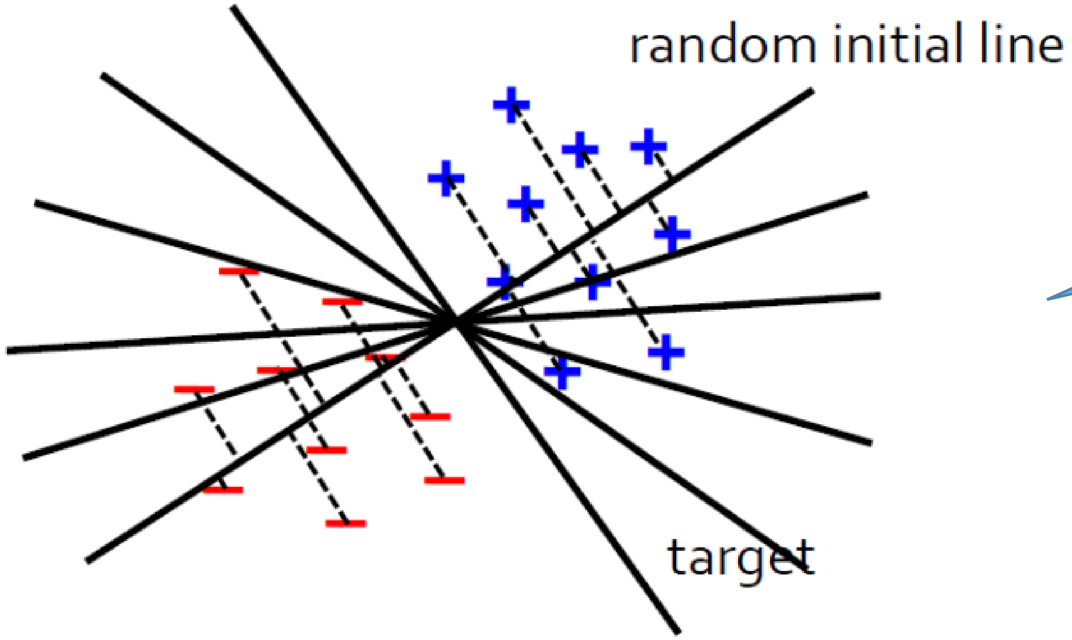
Label

1: about science
0: not about science

CTR, inches of rainfall, ...

Logistic Regression

Goal : Find best line separating two sets of points.



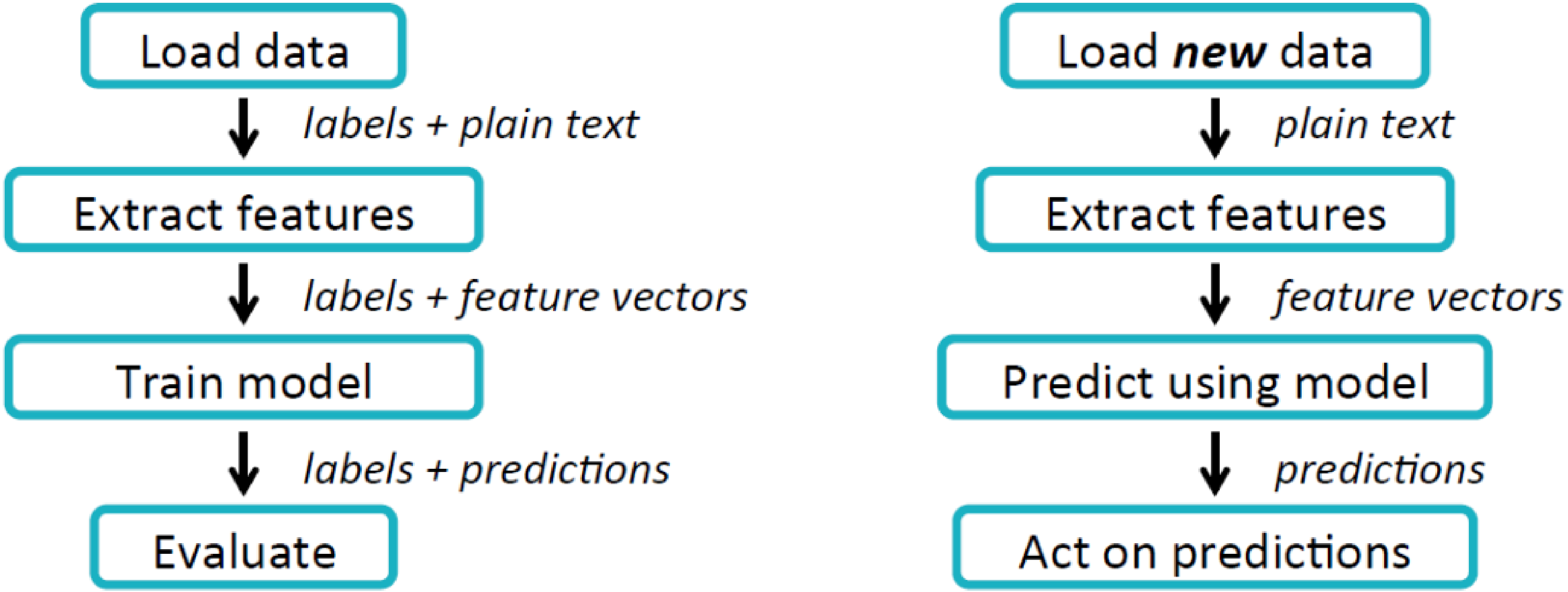
Classify into science and non-science. Each point represents a document.

[Further details on logistic regression can be found at - https://en.wikipedia.org/wiki/Logistic_regression](https://en.wikipedia.org/wiki/Logistic_regression)

Machine Learning Workflow and Challenges

Example ML Workflow

TRAINING **TESTING/PRODUCTION**



Almost identical workflow

What are the pain points?

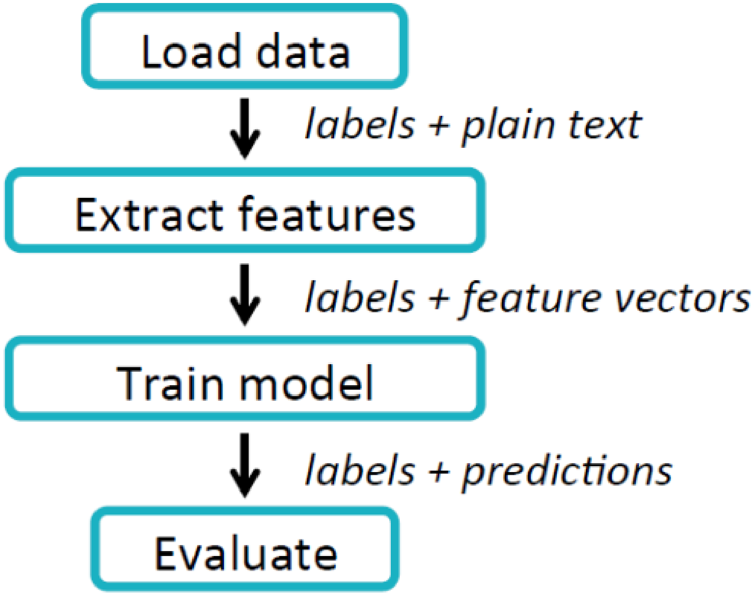
- Create and Handle Many RDDs and data types
  - Labels, features, predictions..
- Write as a script
  - Whole pipeline needs to be coded as a script
  - Not modular
- Tune parameters
  - Key part of ML
  - Training many models
    - For different splits of data
    - Different sets of parameters

Solving the Machine learning challenges

How can the task be made easier?

- Make RDDs easier to read
  - Have to explicitly break up the fields in RDD
  - E.g., break line into blank separated tokens
- As developers we would like to just
  - Program to extract features
  - Specify the model to be used
- However, ML needs additional work
  - Write a script to do all the steps
  - Train the model
  - Evaluate the error of the model by testing

TRAINING



Key concepts

- Reading RDDs: DataFrame

Solves the RDD creation pain-point

- ML Pipeline
    - Transformers
    - Estimators
    - Evaluators

Solves the Scripting..

- Parameters
    - API
    - Tuning

Solves the parameter tuning pain point

- Recall
- Announced Feb 2015
- Inspired by data frames in R and Pandas in Python
- Works in:

http://training.databricks.com/intro.pdf

What is a Dataframe?

- a distributed collection of data organized into named columns
- Like a table in a relational database

Dataframes

Features

- Scales from KBs to PBs
- Supports wide array of data formats and storage systems (Hive, existing RDDs, etc)
- State-of-the-art optimization and code generation via Spark SQL Catalyst optimizer
- APIs in Python, Java

Dataframe : RDD + Schema + DSL

Named columns with types

```
label: Double
text: String
words: Seq[String]
features: Vector
prediction: Double
```

label	text	words	features
0	This is ...	[“This”, “is”, ...]	[0.5, 1.2, ...]
0	When we ...	[“When”, ...]	[1.9, -0.8, ...]
1	Knuth was ...	[“Knuth”, ...]	[0.0, 8.7, ...]
0	Or you ...	[“Or”, “you”, ...]	[0.1, -0.6, ...]

Domain-Specific Language

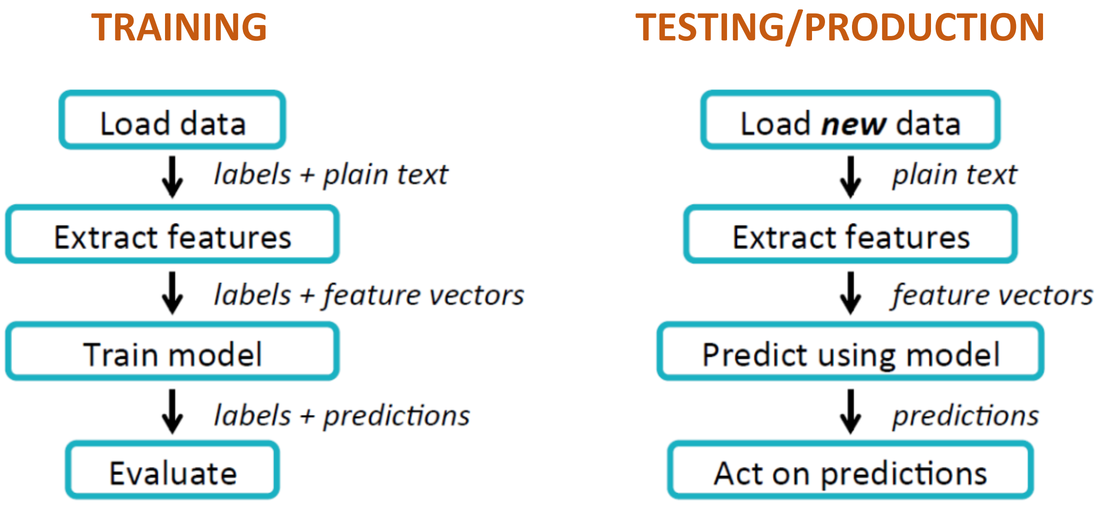
```
# Select science articles
sciDocs =
  data.filter(“label” == 1)

# Scale labels
data(“label”) * 0.5
```

ML Pipelines

ML Pipelines

- Introduced in Spark 1.2 and 1.3
- Allows developers to just
    - Program to extract features
    - Specify the model to be used
- Automates the process of
    - Write a script to do all the steps
    - Train the model
    - Evaluate the error of the model by testing
    - Or deploy in production



The ML Pipeline

Transformers

- Extract features from DataFrame
- Features are stored in a new DataFrame

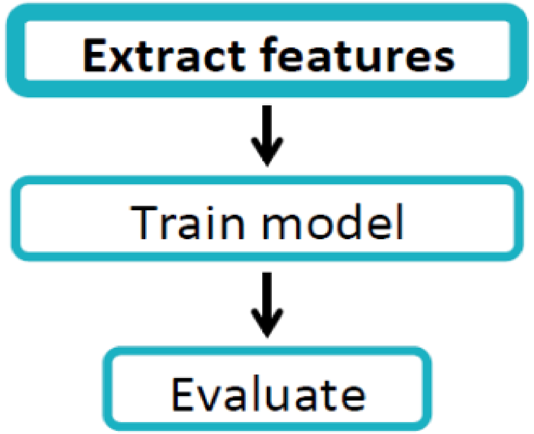
Estimators

- ML Algorithms
- MLLib has standard defined ML algorithms (e.g., Logistic Regression)
- User can add his own

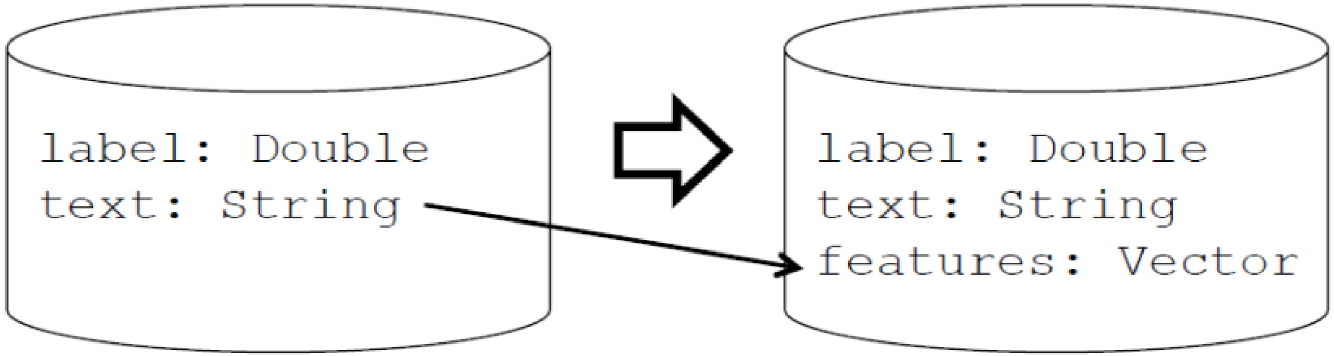
Evaluators

- Compute predictions and estimate metrics such as error
- Tune algorithm parameters
- Evaluator depends upon estimator
  - Evaluator that trains Logistic Regression cannot be used for Decision Trees

TRAINING



`def transform(DataFrame): DataFrame`



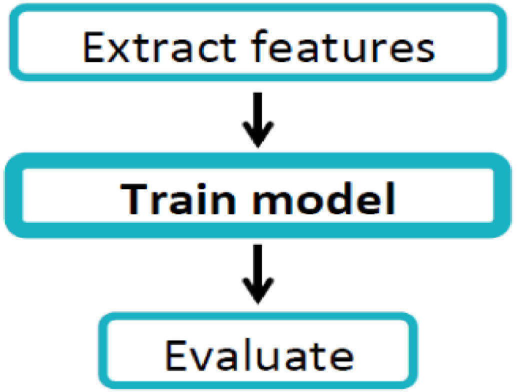
Label	Text
0	
1	

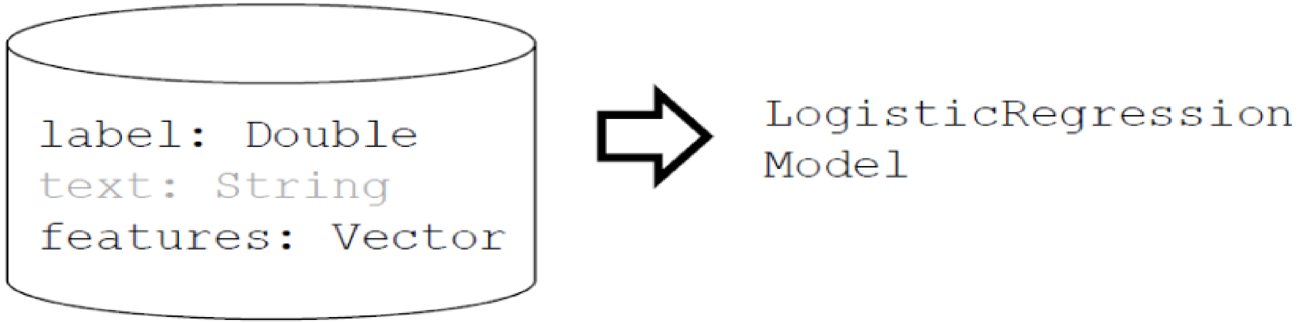
Label	Text	Features
0		
1		

Estimator

TRAINING

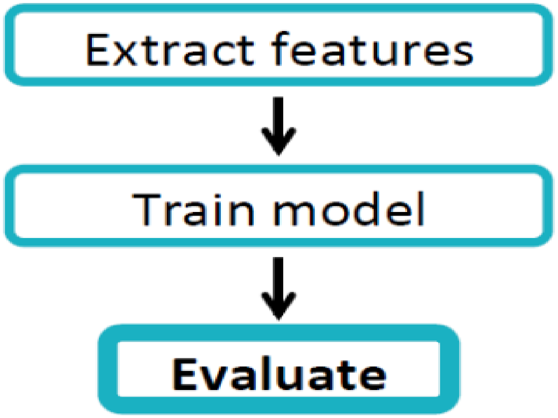
`def fit(DataFrame): Model`



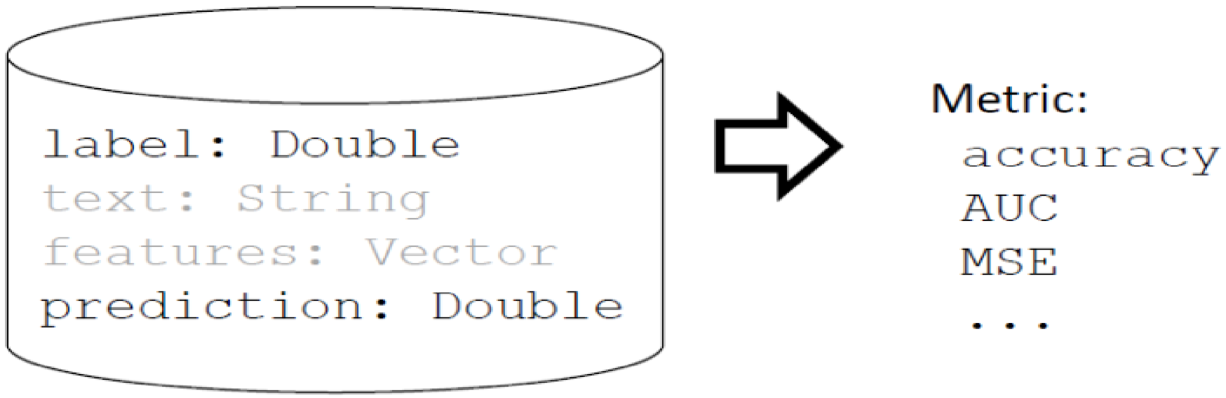


Label	Text	Features
0		
1		

TRAINING



`def evaluate(DataFrame): Double`



Label	Text	Features	Prediction
0			
1			

Model

TESTING/PRODUCTION

Extract features

↓

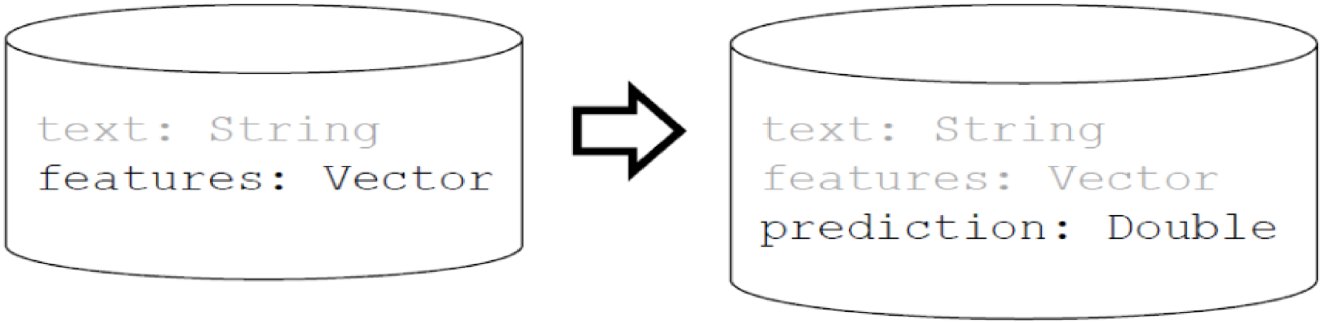
Predict using model

↓

Act on predictions

Model is a type of Transformer

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

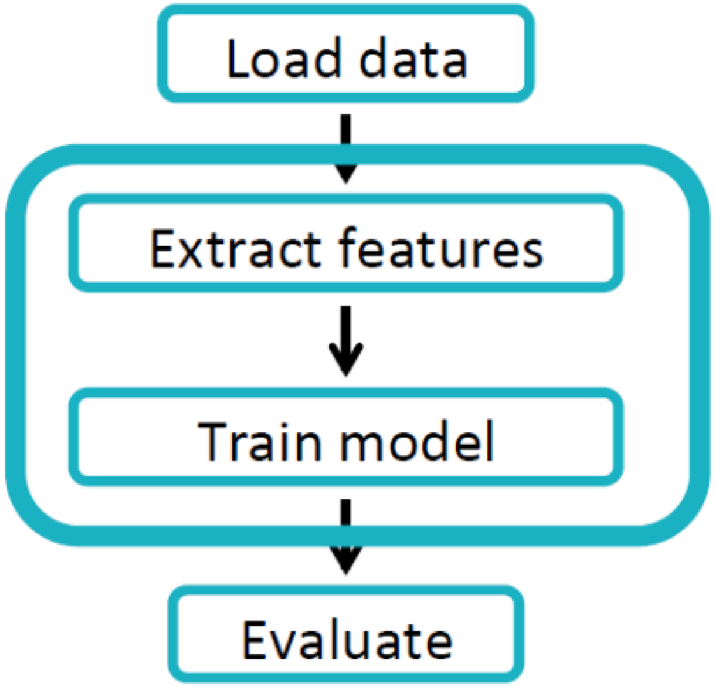


Text	Features

Text	Features	Prediction

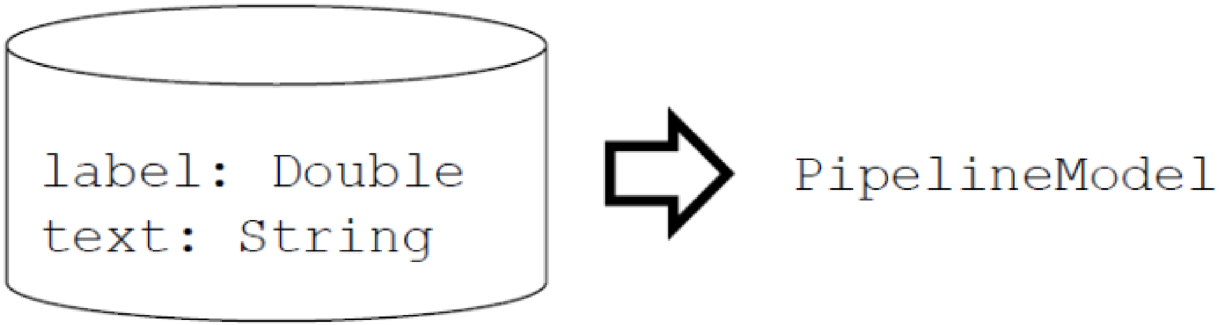
The training Pipeline

TRAINING



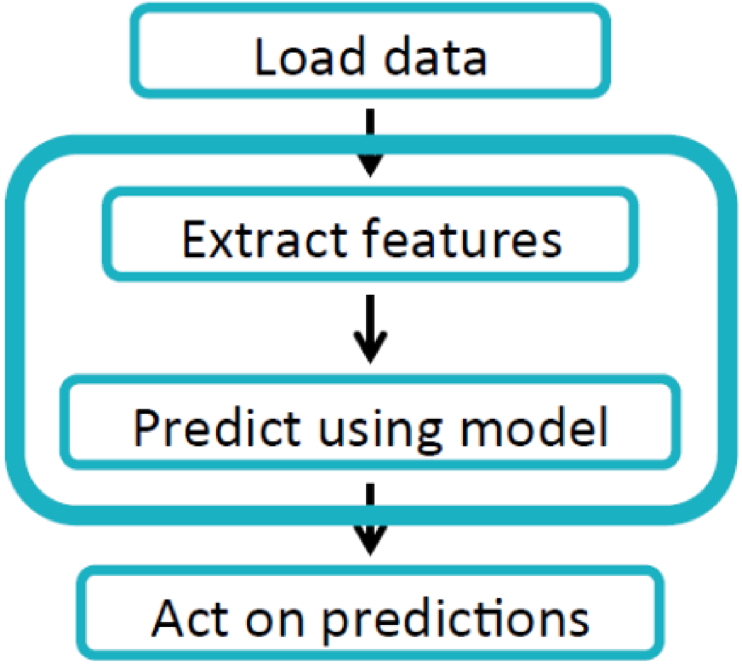
Pipeline is a type of Estimator

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



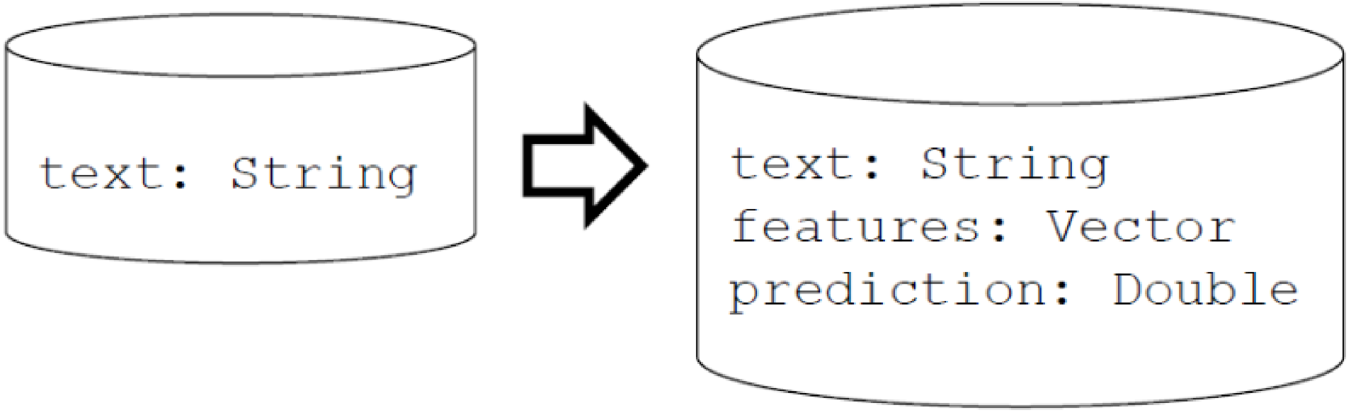
The testing pipeline/model

TESTING/PRODUCTION

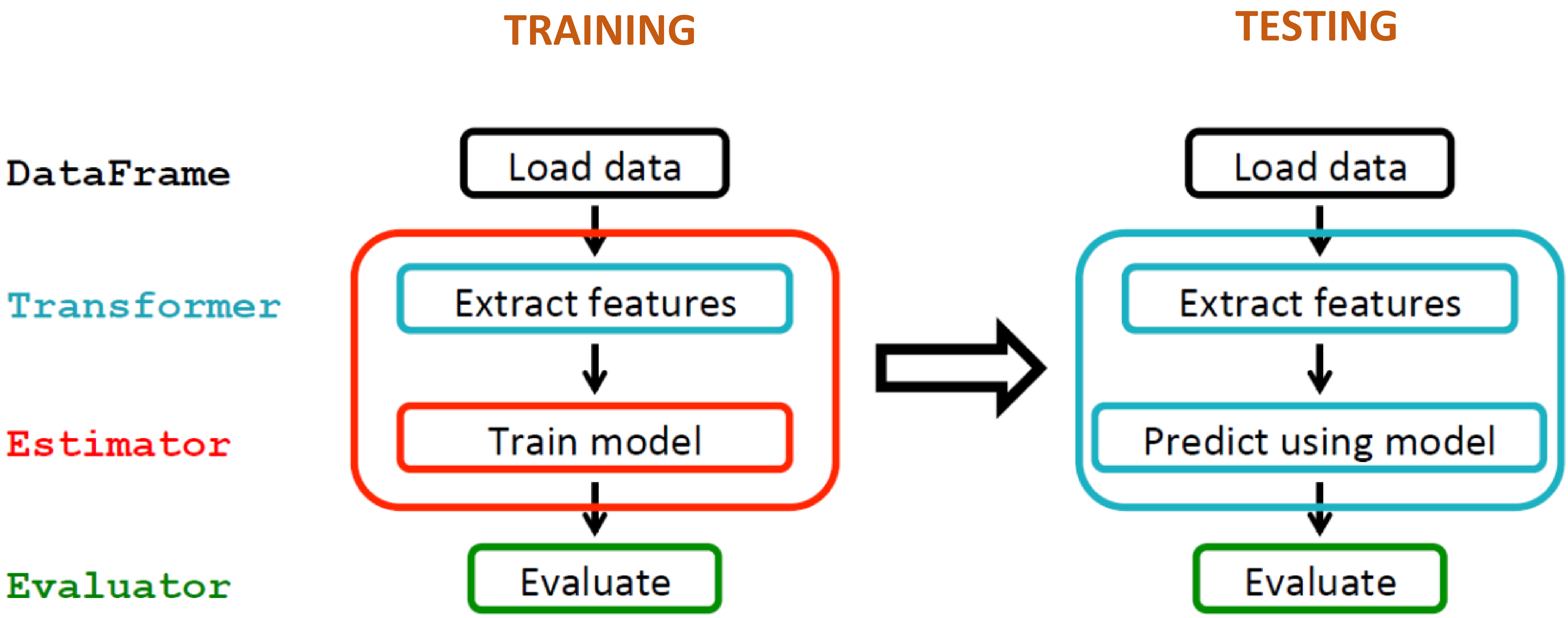


PipelineModel is a type of Transformer

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



Putting it all together



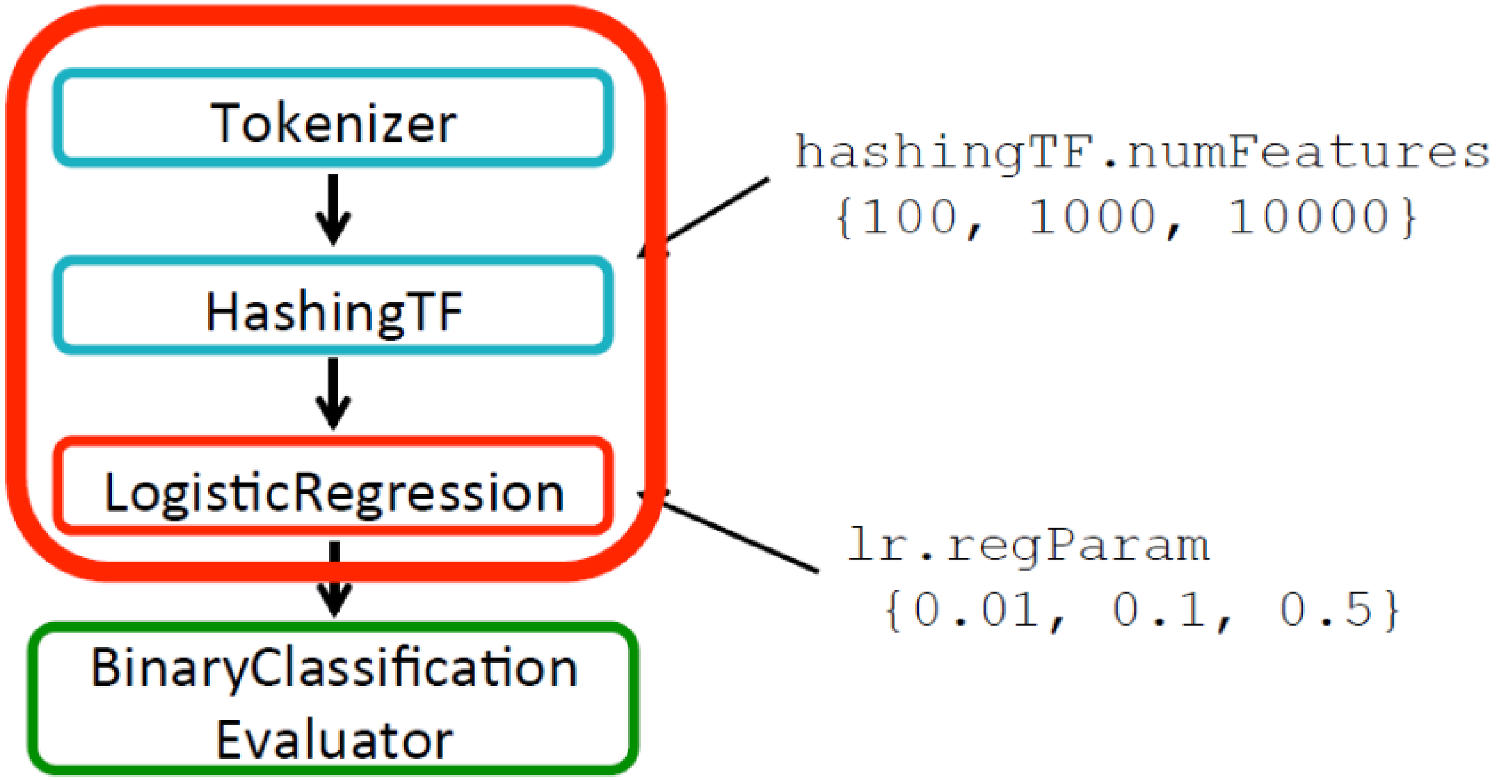
Parameter Tuning

Given:

- Estimator
- Parameter Grid
- Evaluator

Find the Best parameters

CrossValidator



Exercise 4: 10 minutes

- 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.

Exercise 4: Solution

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

- Input DataFrame
  - <bell.wav>, bell
- Feature DataFrame
  - <bell.wav>, bell, frequencies
- Transformer (use same transformer for train/ predict)
  - <bell.wav> Bell, average frequency
- Model
  - train(FeatureDataFrame)
    - Associate average frequency for “Bell”
  - predict(PredictDataFrame)
    - Output closest matching sound

Mllib algorithms

- Classification
  - Logistic Regression
  - Decision Tree
  - Random Forest
  - Gradient boosted tree
  - Multilayer Perceptron
  - SVM
  - Naïve Bayes
- Clustering
  - K-Means
  - LDA
  - GMM
- Collaborative Filtering
  - ALS
- Frequent Pattern Mining

Deep Learning with Big Data

Challenges for Deep Learning

- Heterogenous cluster
- Deep Learning (Tensorflow)
  - Iterative
  - Matrix vector multiplication – Linear algebra
- Initially evolved on a single machine – only scale up
- Then had its own cluster
  - Typically heterogenous with CPUs, GPUs, TPUs

Challenges for Deep Learning

- But data resides on HDFS and big data platform uses Spark
- How should the two work together.
- Typically the two clusters are different

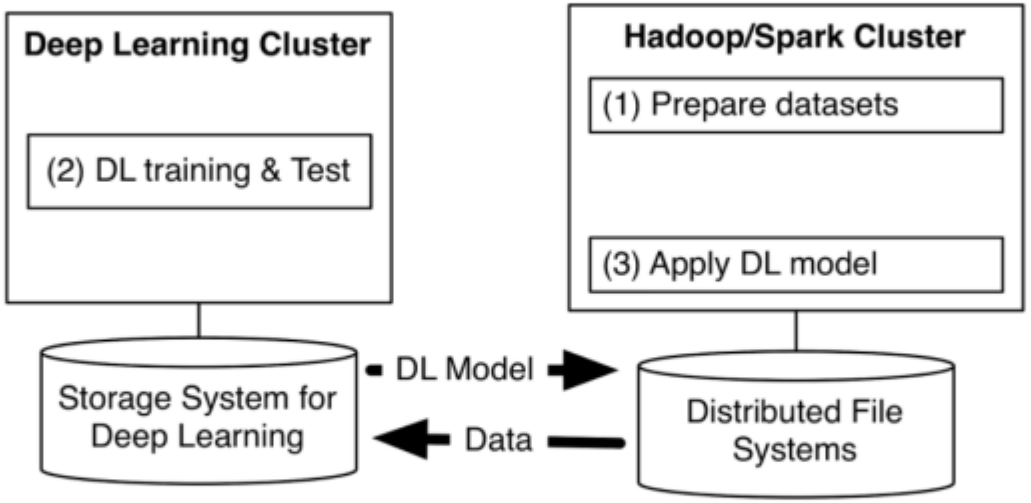


Figure 1: ML Pipeline with multiple programs on separated clusters

https://developer.yahoo.com/blogs/157196317141/

Challenges for Deep Learning

- Can we use the same cluster?
- Tensorflow on Spark
    - From Yahoo

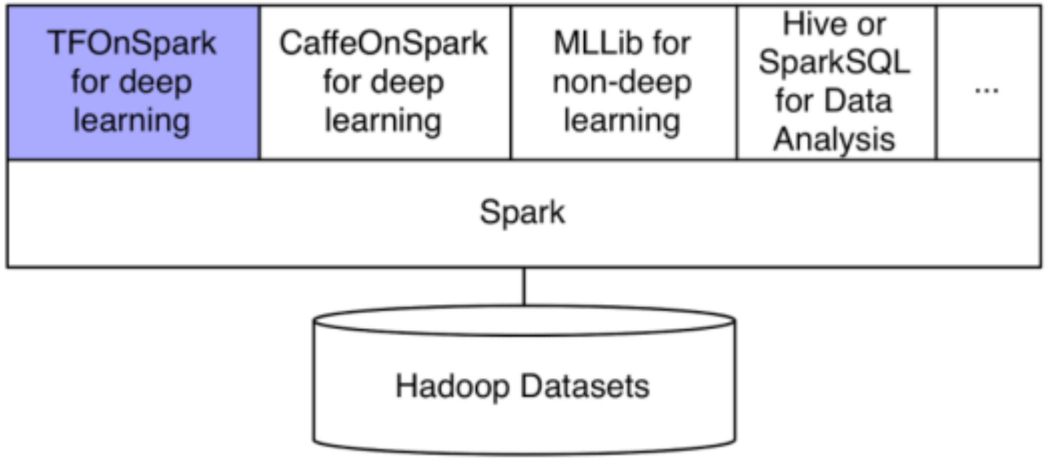


Figure 2: TensorFlowOnSpark for deep learning on Spark clusters

https://developer.yahoo.com/blogs/157196317141/

Tensorflow on Spark Architecture

- Supports both
  - Model parallelism
  - Data parallelism
- <10 lines of code change reqd
- Algorithm and parameter server run on Spark executors
  - Can read data directly from HDFS
  - Spark RDD data is fed to spark executor which passes it to Tensorflow
- RDMA → faster network transfers

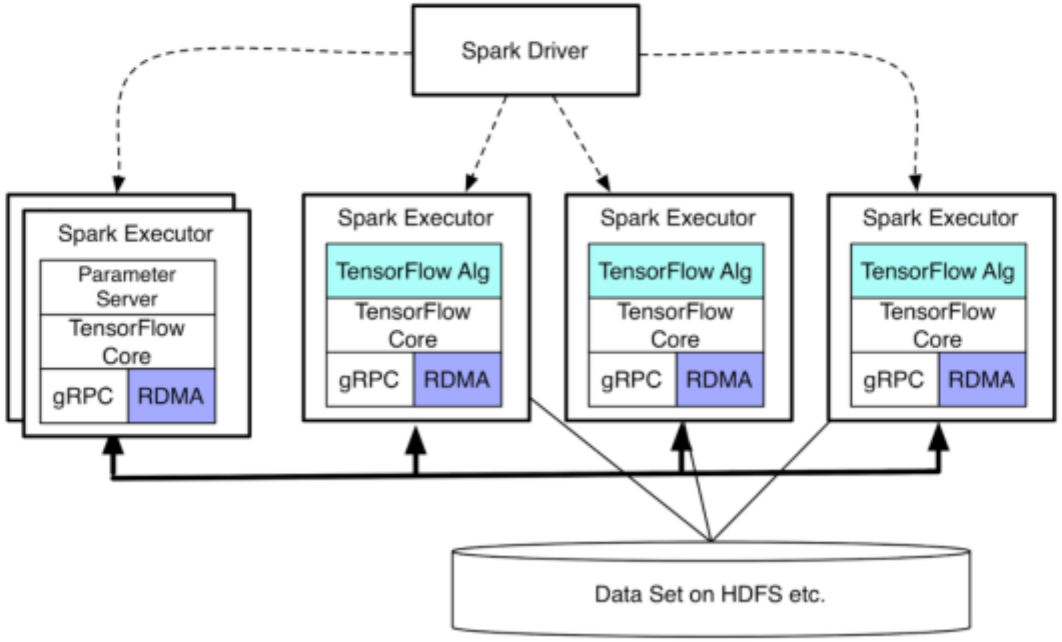


Figure 3: TensorFlowOnSpark system architecture

https://developer.yahoo.com/blogs/157196317141/

Other solutions

- SystemML – systemml.apache.org
  - IBM
  - SystemML: Declarative Machine Learning on Spark http://www.vldb.org/pvldb/vol9/p1425-boehm.pdf
  - Uses a declarative ML language
  - Translated to MR/Spark
- Intel BigDL
  - Modeled on Torch

THANK YOU

K V Subramaniam, Usha Devi
Dept. of Computer Science and Engineering

subramaniamkv@pes.edu
ushadevibg@pes.edu