MLlib and GraphX

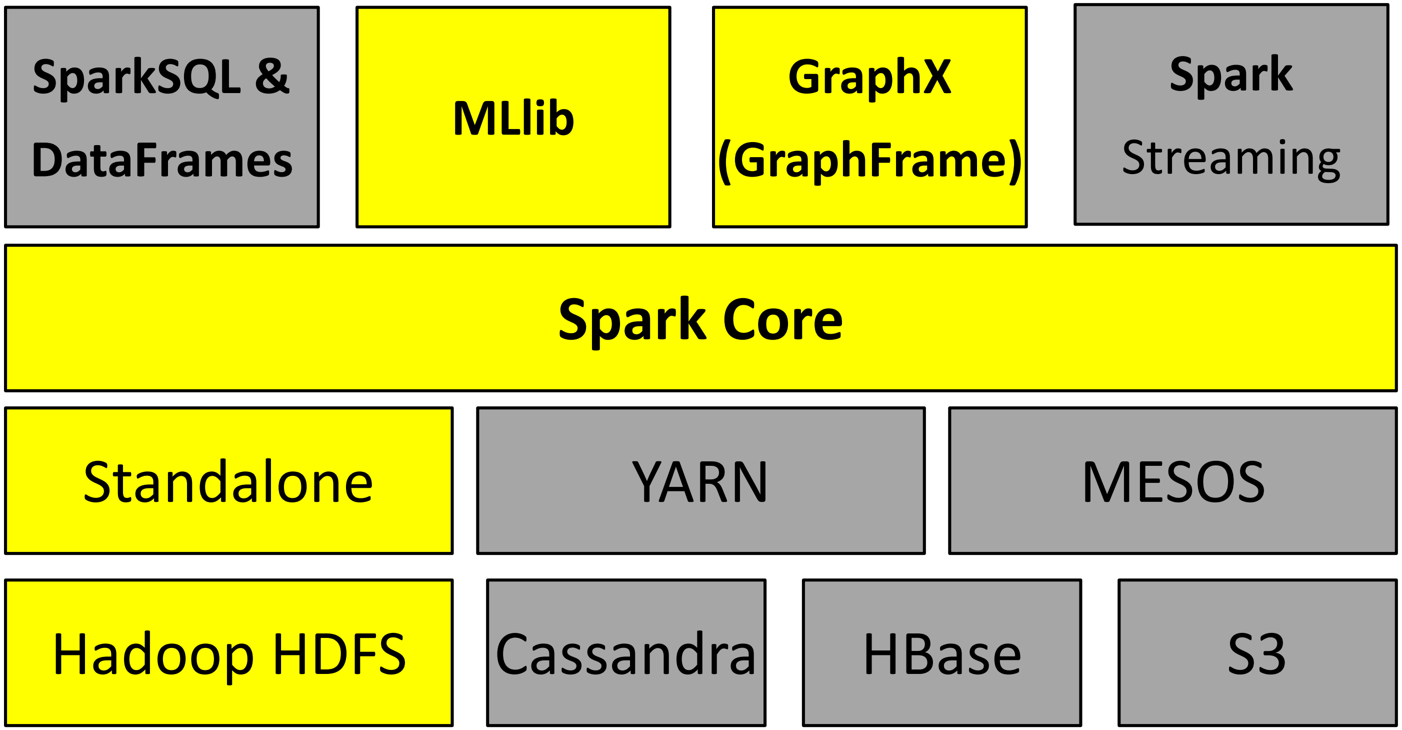
Lecture 6
November 15th, 2017

Jonghyun Bae (jonghbae@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: Holden Karau et al. (Learning Spark), Xiangrui Meng (MLlib), Joseph Bradley (Machine Learning Model Persistence), Jae W. Lee (SSE2029/2016-2)*

MLlib & GraphFrames

- **Special-purpose libraries for a variety of data science tasks**
  - MLlib
  - GraphX



*** Image from https://www.safaribooksonline.com/library/view/data-analytics-with/9781491913734/ch04.html**

Outline

- **Machine learning**
- Basics of MLlib
- Representative algorithms with MLlib
- Graph theory
- Graph structure in Spark (GraphX)
- Graph Algorithms with GraphX

Machine learning

- **ML is everywhere**

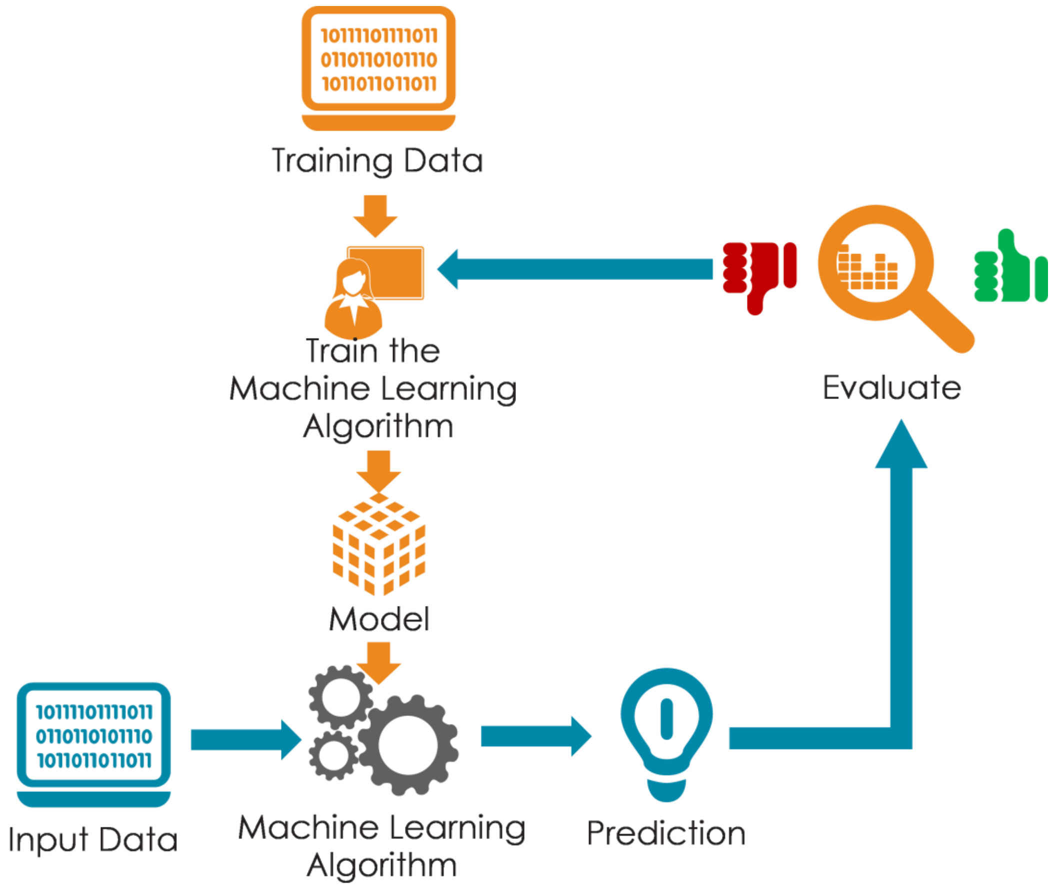
What is machine learning?

- **Machine is learning itself! (So simple!)**

- **Learning the machine from the data and executing the operation that is not specified by the human being (complex mean)**

Principle of machine learning

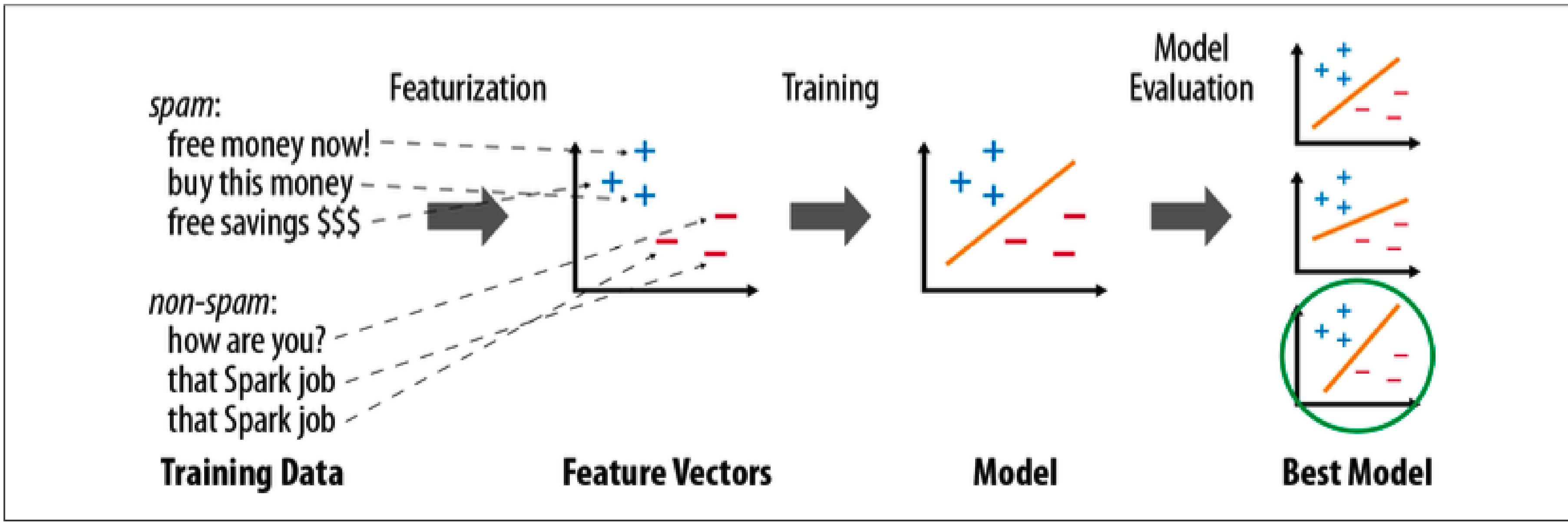
- Recognition – Training – Evaluation – Recognition (repeat)



*** Image from http://blogs.teradata.com/data-points/building-machine-learning-infrastructure-2/**

Logistic regression example

- Recognition (Featurization)
- Training
- Evaluation



Machine learning의 방법

- **Supervised learning (지도 학습 in Kor.)**
  - Inferring a function from labeled training data
  - Decision tree, random forest, linear regression, naïve bayesian,…

- **Unsupervised learning (자율 학습 in Kor.)**
  - Inferring a function to describe hidden structure from **unlabled data**

ML and data mining

- **Machine learning: Find predicted results from known models**
- **Data mining: Find unexpected results from unclustered data**

Outline

- Machine learning
- **Basics of MLlib**
- Representative algorithms with MLlib
- Graph theory
- Graph structure in Spark (GraphX)
- Graph Algorithms with GraphX

What is MLlib*

- Spark's library of machine learning functions
- Make practical machine learning scalable and easy
- Good combination with RDD persist (`*.cache()`)
  - Lots of ML is iterative algorithm

* X. Meng et al. MLlib: Machine Learning in Apache Spark, JMLR, 17(34):1–7, 2016.

Benefits of MLlib

- **Large dataset learning is possible using MapReduce method**

- **Algorithm implemented in consideration of parallel environment**

RDD-based and DataFrame-based

- **Why DataFrame-based API is recommended?**

  - More user-friendly and understandable API than RDDs

  - Uniform API across ML algorithms and across multiple languages

Data type of MLlib

- **Vector**
  - Mathematical vector.
  - Dense vector (all entry is stored) and Sparse vector (non-zero is stored only)
- **LabledPoint**
  - For supervised learning algorithms such as classification and regression
- **Rating**
  - A rating of a produce by a user, used in recommendation

Basics of MLlib

- **High-level tools provided such as**
  - ML Algorithms: common learning algorithms
    - Classification, regression, clustering and collaborative filtering
  - Featurization: feature extraction, transformation, dimensionality reduction
  - Pipelines: tools for constructing, evaluating, and tuning ML Pipelines

Basics of MLlib

- **High-level tools provided such as**
  - Persistence: saving and load algorithms, models, and Pipelines
  - Utilities: linear algebra, statistics, data handling

Outline

- Machine learning
- Basics of MLlib
- **Representative algorithms with MLlib**
- Graph theory
- Graph structure in Spark (GraphX)
- Graph Algorithms with GraphX

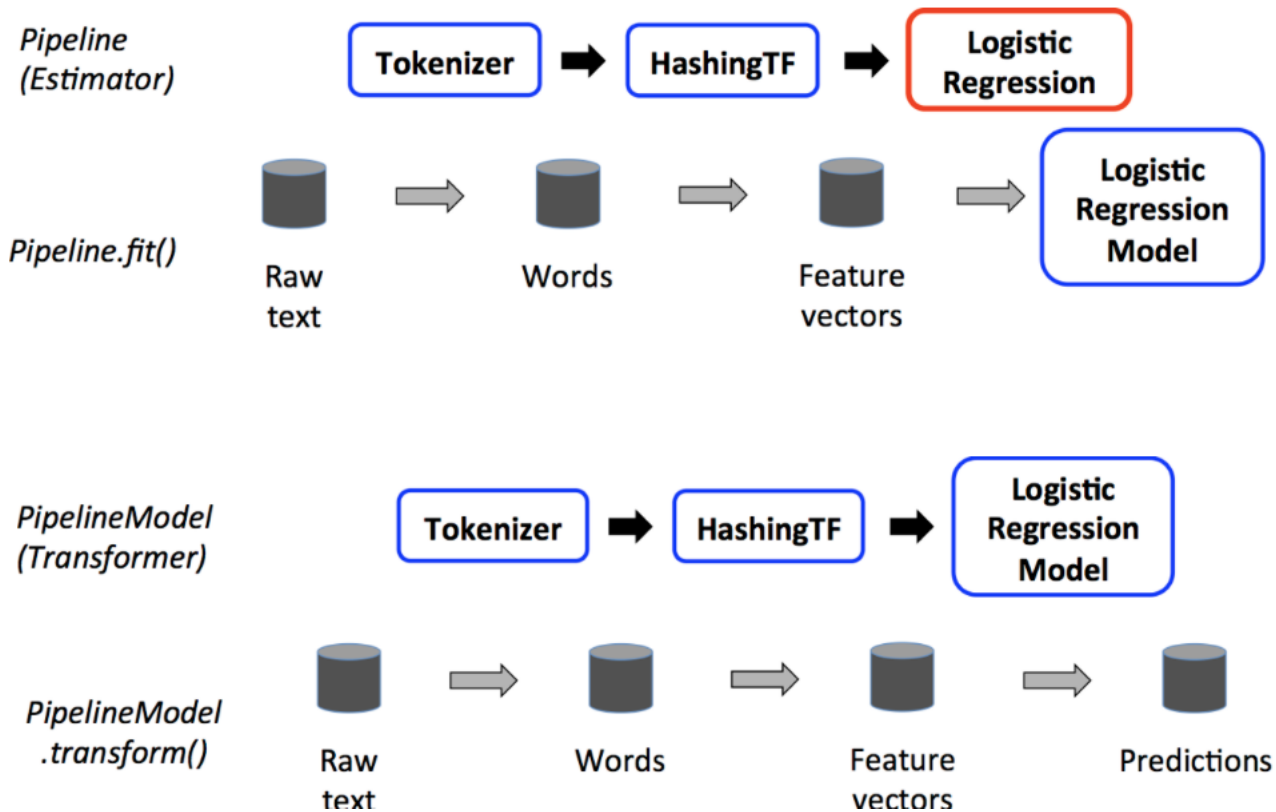
Pipelines

- **Purpose to make it easier to combine multiple algorithms into a single pipeline**

- **Transformers**
  - Abstraction that include feature transformers and learned models

- **Estimators**
  - Abstracts the concept of a learning or any algorithm that fits or trains on data

Pipelines example



Pipeline (Estimator)

```
>>> training = spark.createDataFrame([
...     (0, "a b c d e spark", 1.0),
...     (1, "b d", 0.0),
...     (2, "spark f g h", 1.0),
...     (3, "hadoop mapreduce", 0.0)
... ], ["id", "text", "label"])

>>> tokenizer = Tokenizer(inputCol="text", outputCol="words")
>>> hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(),
outputCol="features")
>>> lr = LogisticRegression(maxIter=10, regParam=0.001)
>>> pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
>>> model = pipeline.fit(training)
```

Pipeline (Transformer)

```
>>> test = spark.createDataFrame([
...      (4, "spark i j k"),
...      (5, "l m n"),
...      (6, "spark hadoop spark"),
...      (7, "apache hadoop")
... ], ["id", "text"])
>>> prediction = model.transform(test)
>>> selected = prediction.select("id", "text", "probability,
prediction")
>>> for row in selected.collect():
...      rid, text, prob, prediction = row
...      print("(%d, %s) --> prob=%s, prediction=%f" %
...             (rid, text, str(prob), prediction))
```

Featurization

- **Extraction**
  - Extracting features from “raw” data
- **Transformation**
  - Scaling, converting, or modifying features
- **Selection**
  - Selecting a subset from a larger set of features

ML algorithms

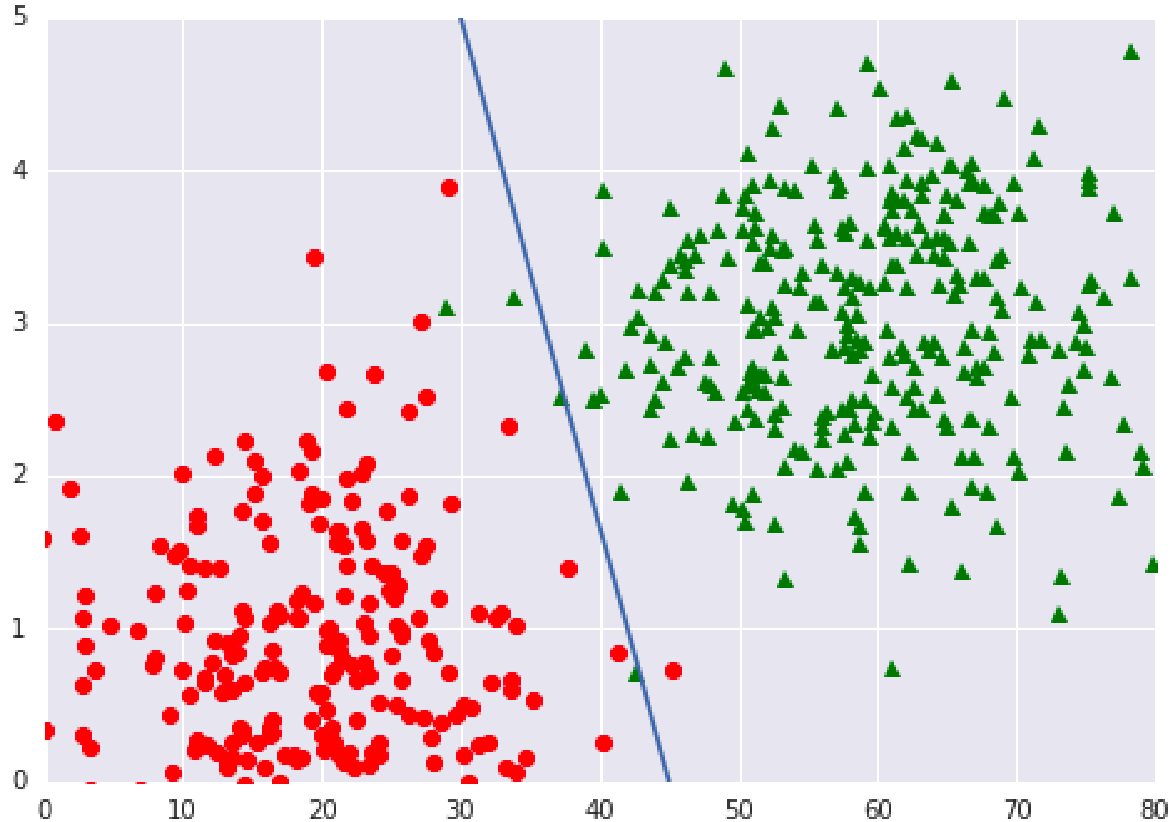
- Classification
- Regression
- Clustering
- Collaborative filter

Classification

- Predict category or class "Y" from some inputs "X"
- Logistic regression, Decision tree classifier, Random forest classifier, Bayesian classification, …

Logistic regression

- Detecting email spam and normal email
- Detecting normal transactions and abnormal transactions in credit card transactions



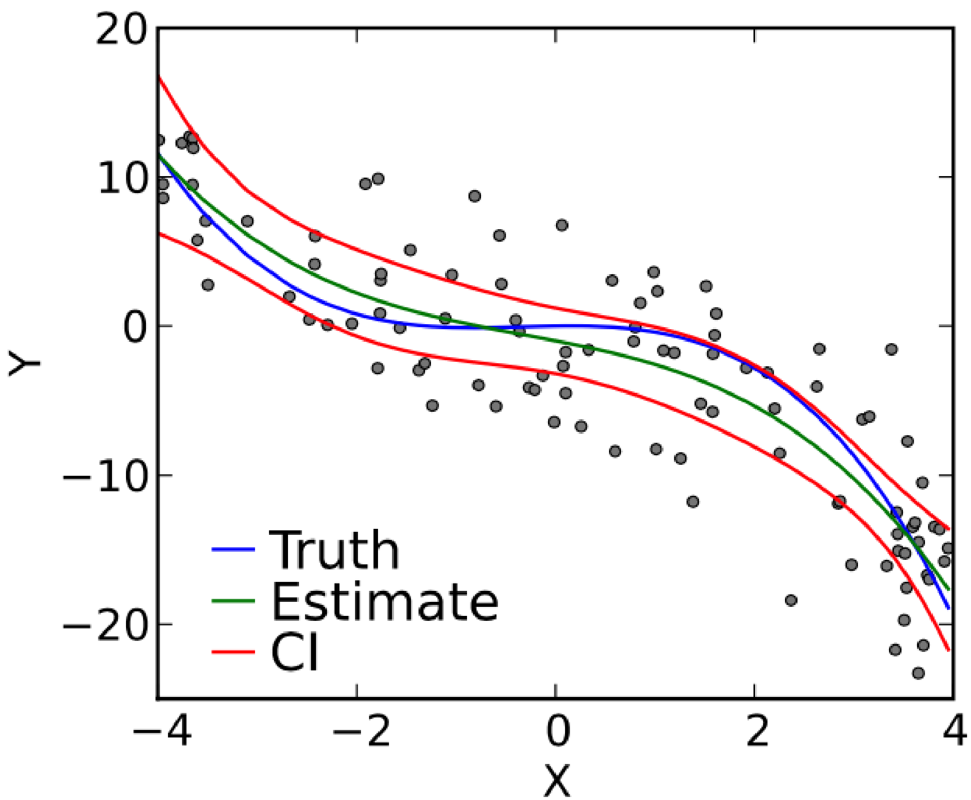
Regression

- Predictions from data by learning the relationship between features of your data and some observed, continuous-valued response

- Linear regression, Decision tree classifier, Random forest regression, …

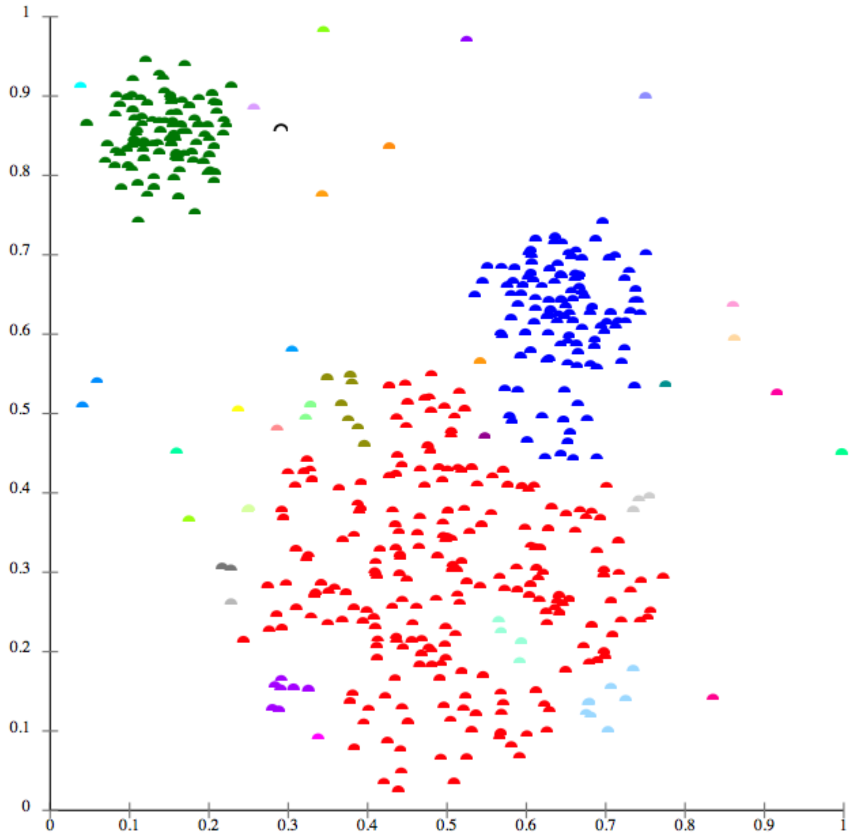
Linear regression

- Linear approach for modeling the relationship between a scala dependent variable Y and one or more variables denoted X



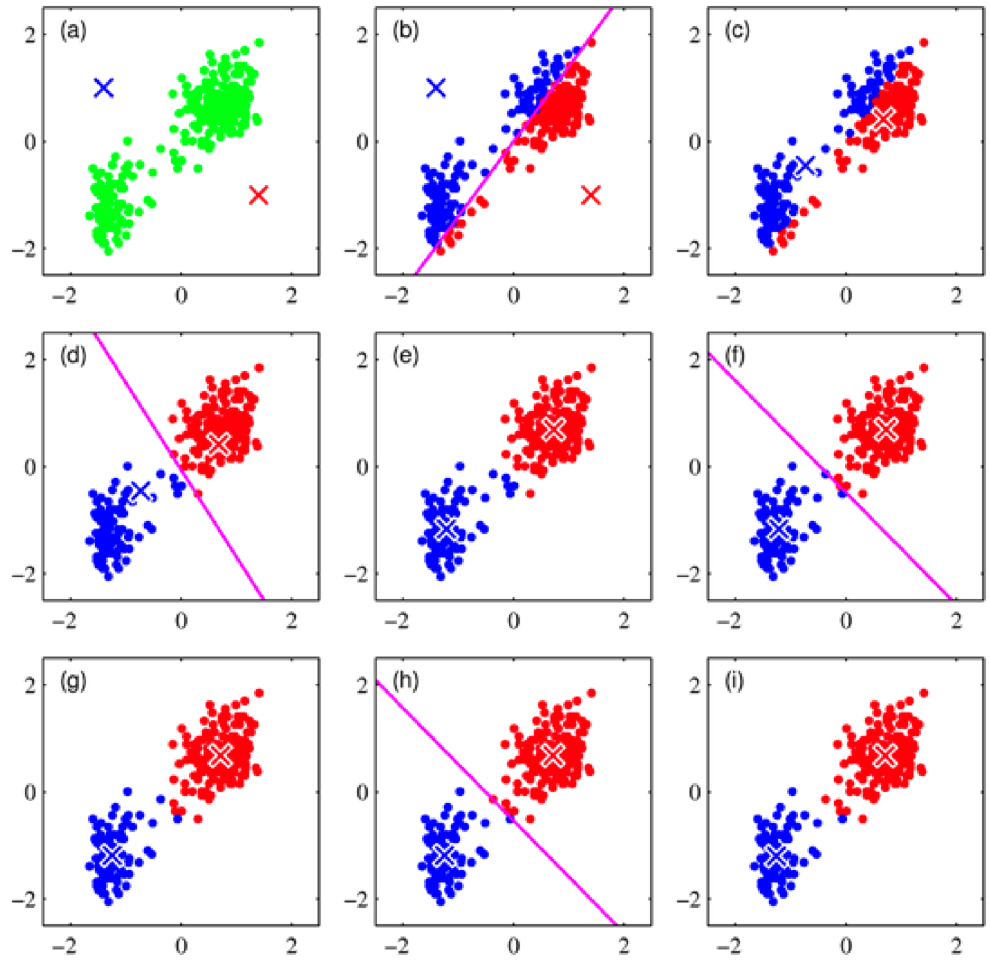
Clustering

- **Assignment of a set of observations into subsets**
  - High similarity between data in clusters



K-means

- Clustering aims to partition n observations into k clusters



Collaborative filter

- Used for recommender system

- Fill in the entries of user-item association matrix

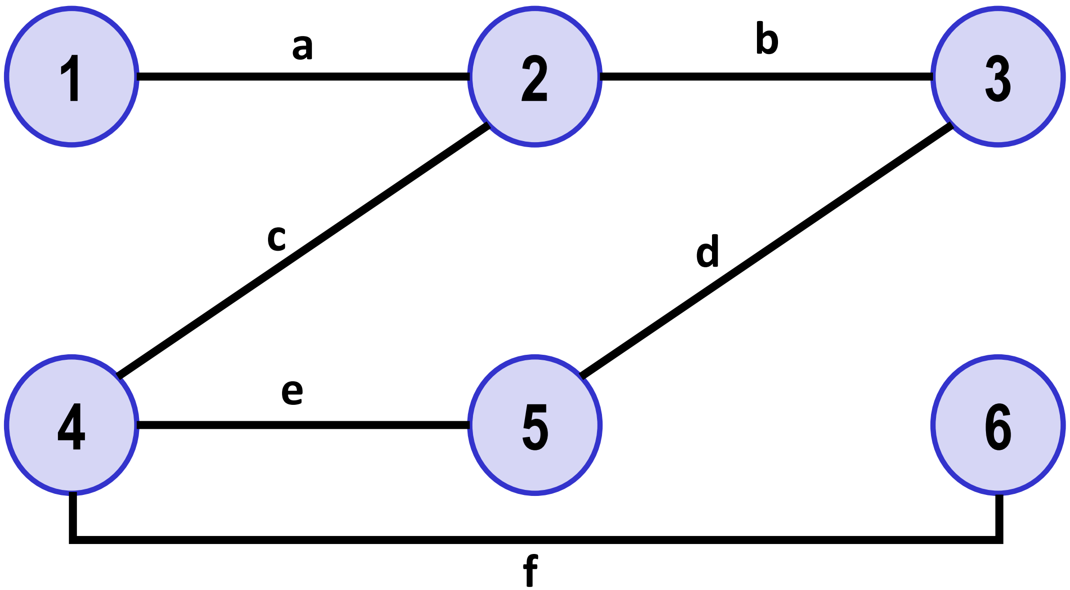
	Alice	Bob	Chaley	Delta	Edgar
The piano	-	-	+		+
Pulp Fiction	-	+	+	-	+
Clueless	+		-	+	-
Cliffhanger	-	-	+	-	+
Fargo	-	+	+	-	? => +

Outline

- Machine learning
- Basics of MLlib
- Representative algorithms with MLlib
- **Graph theory**
- Graph structure in Spark (GraphX)
- Graph Algorithms with GraphX

Graph theory

- Mathematical structures used to model pairwise relations between object
- Made up of vertices which are connected by edges



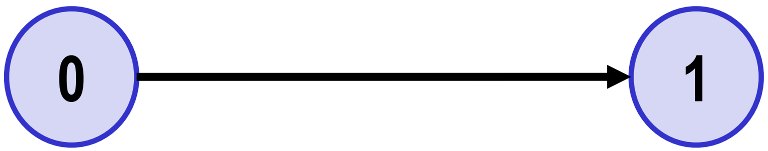
Graph Definitions

- **A graph G consists of two sets: G(V, E)**
- **Edge**
  - Connection between vertices
  - Possible empty set of edges E(G)

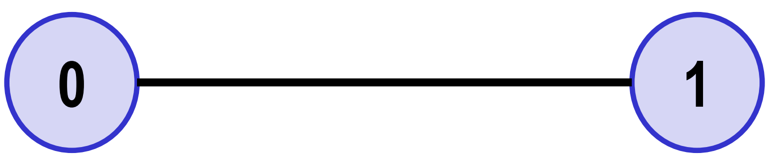
- **Vertex**
  - Fundamental unit of which graphs are formed
  - Nonempty set of vertices V(G}

Graph Definitions

- **Directed graph**
  - Which edge is a directed pair of vertices, $<v_0, v_1>$
  - This is different from $<v_1, v_0>$

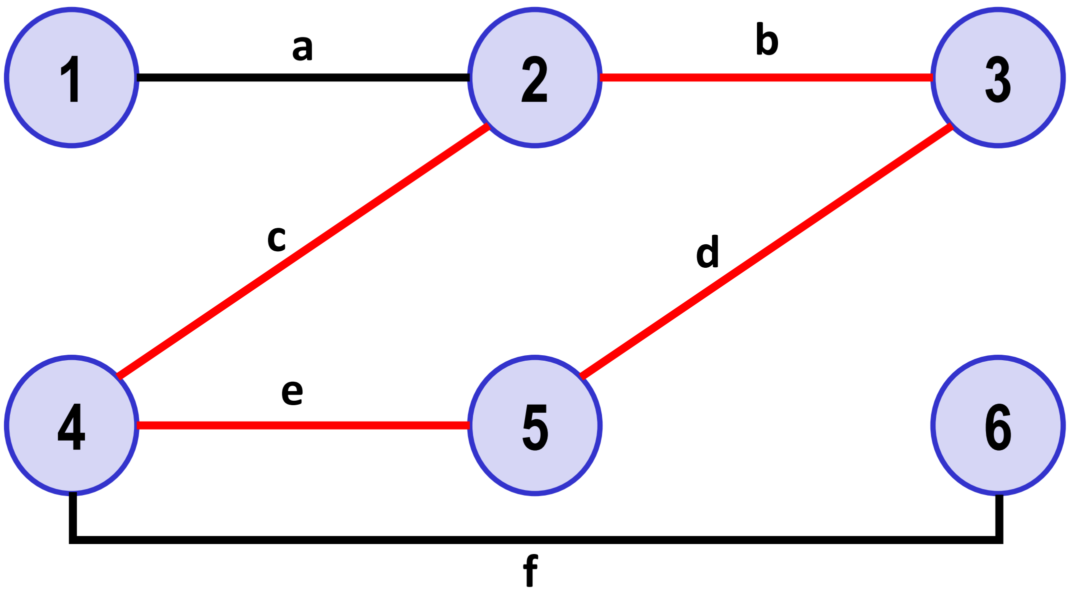


- **Undirected graph**
  - Which the pair of vertices in an edge is unordered
  - $<v_0, v_1> = <v_1, v_0>$



Graph Definitions

- Cyclic / Acyclic graph



- Spark has Directed Acyclic Graph scheduler (DAG scheduler)

Usage of graph algorithm

- 집단 혹은 계층이나 역활의 분화 설명

- 통계적인 모델을 이용해 관계의 영향력이 특정한 결과에 영향을 주는지를 측정

Graph problem example

- **Konigsberg bridge problem**
  - Degree of a vertex: the number of edges incident to it
  - Eulerian walk: a walk starting at a vertex, going through each edge exactly once and terminating at the start vertex

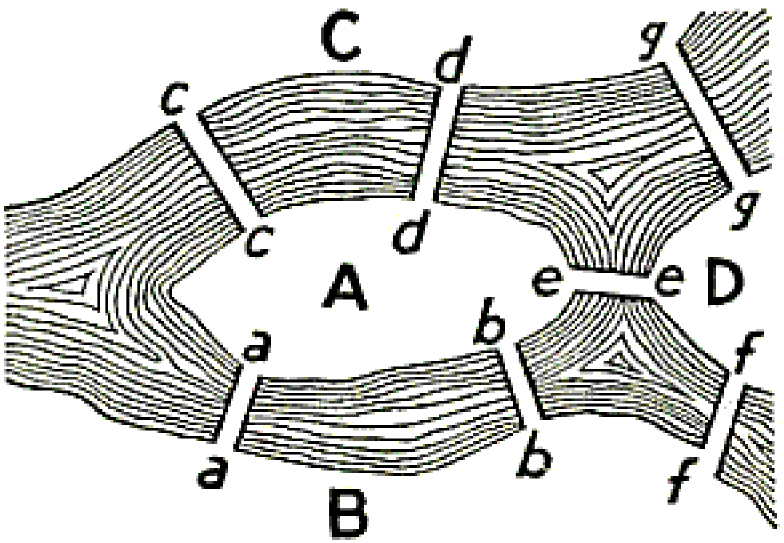
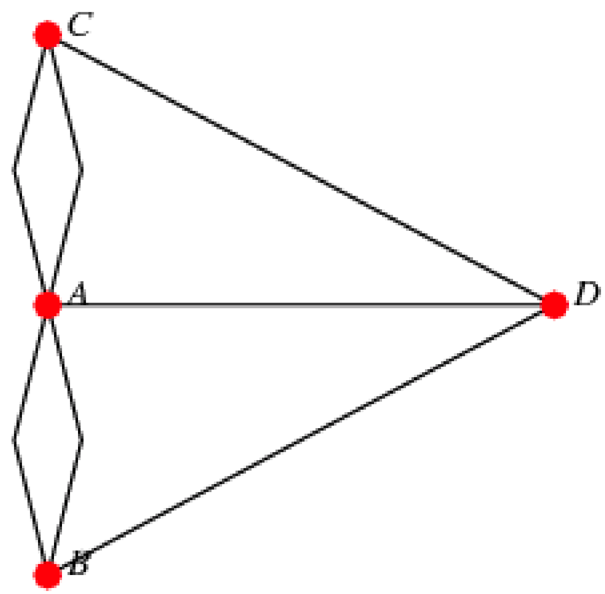


FIGURE 98. *Geographic Map: The Königsberg Bridges.*



*** Image from http://mathworld.wolfram.com/KoenigsbergBridgeProblem.html**

Outline

- Machine learning
- Basics of MLlib
- Representative algorithms with MLlib
- Graph theory
- **Graph structure in Spark (GraphX)**
- Graph Algorithms with GraphX

GraphX

- Graph-parallel computation

- Optimize the representation of vertex and edge types reducing the in memory footprint

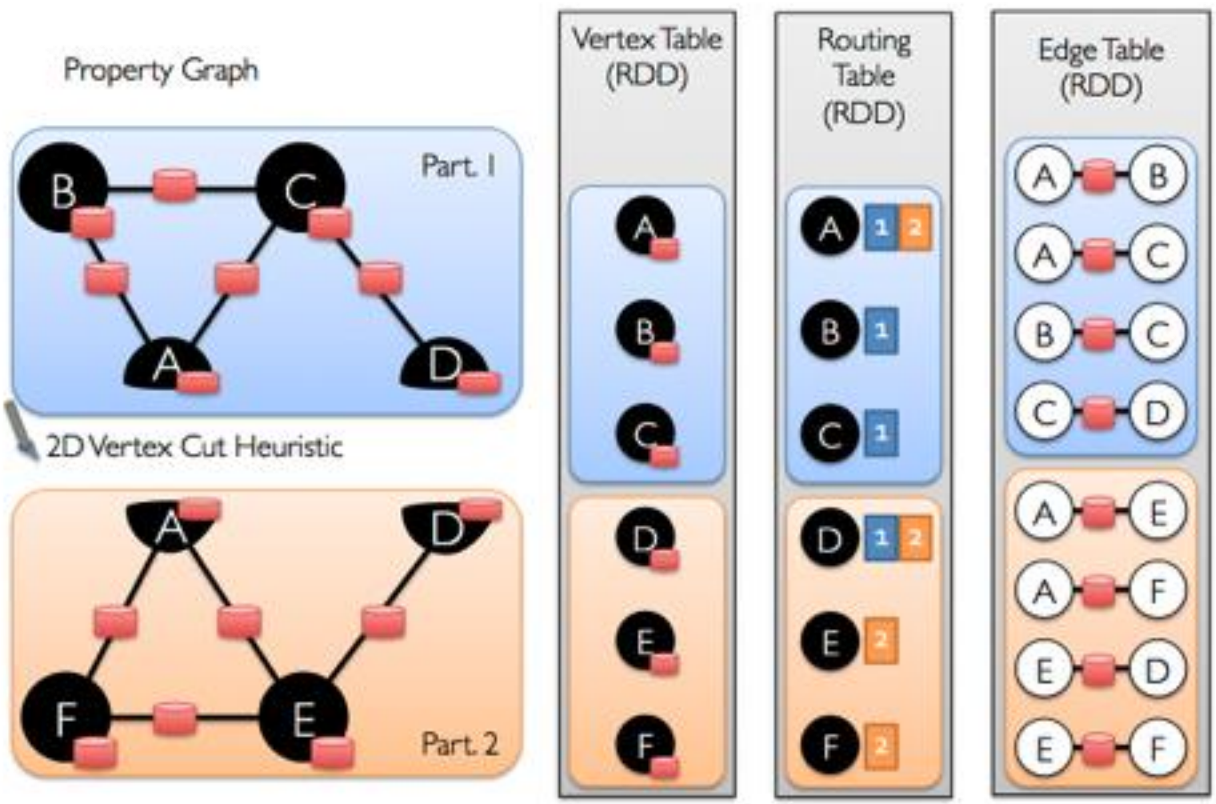
Graph structure in Spark

- **VertexRDD**
  - `RDD[(VertexID, A)]` and adds the additional constraint that each `VertexId` occurs only once
  - Set of vertices each with an attribute of type `A`
  - `filter, minus, diff, innerJoin`,…

- **EdgeRDD**
  - `RDD[Edge[ED]]` organized the edges in blocks partitioned
  - `mapValues, reverse, innerJoin`

Graph structure in Spark

- **Optimized partitioning strategy**



GraphFrame

- Graph structure based on DataFrame

- Vertex DataFrame

- Edge DataFrame

Example of GraphFrame

- **Spark shell with GraphFrames use a specific version of the GraphFrames package**

```
./bin/pyspark --packages graphframes:graphframes:0.5.0-spark2.1-
s_2.11
```

Example of GraphFrame

```
>>> v = sqlContext.createDataFrame([
...    ("a", "Alice", 34),
...    ("b", "Bob", 36),
...    ("c", "Charlie", 30),
...    ("d", "David", 29),
... ], ["id", "name", "age"])
>>>
>>> e = sqlContext.createDataFrame([
...    ("a", "b", "friend"),
...    ("b", "c", "follow"),
...    ("c", "b", "follow"),
...    ("d", "a", "friend"),
... ], ["src", "dst", "relationship"])
```

Example of GraphFrame

```
>>> from graphframes import *
>>> g = GraphFrame(v, e)
# Query: Get in-degree of each vertex.
>>> g.inDegrees.show()
+---+--------+
| id|inDegree|
+---+--------+
|  c|       1|
|  b|       2|
+---+--------+
>>>
```

Outline

- Machine learning
- Basics of MLlib
- Representative algorithms with MLlib
- Graph theory
- Graph structure in Spark (GraphX)
- **Graph Algorithms with GraphX**

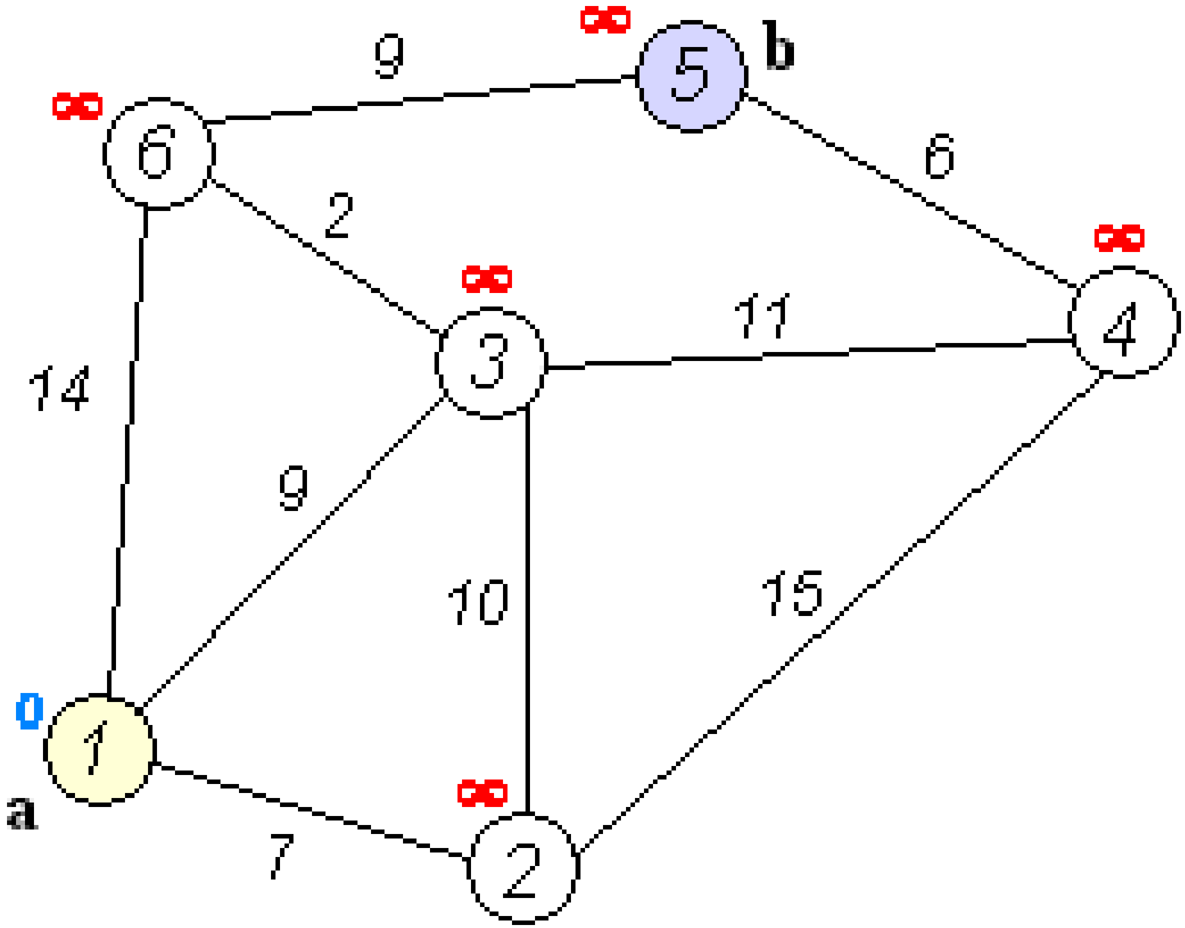
Shortest path

- **Based on Dijkstra's algorithm**
- **Fine the min cost of the path from a given source node to every other node**
- **Given**
  - The cost $e(v_i, v_j)$ of all edges
  - v_0 is the source node
  - $e(v_i, v_j)$ = infinite if v_i and v_j are not connected

Shortest path pseudo code

```
S <- {v0};
dist[v0] <- 0;
for each v in V - {v0} do dist[v] <- e(v0, v);
while S != V do
    choose a vertex w in V-S such that disk[w] is a minimum;
    add w to S;
    for each v in V-S do
        disk[v] <- min(dist[v], dist[w] + e(w, v));
```

Shortest path example



* **Image from https://commons.wikimedia.org/wiki/File:Dijkstra_Animation.gif**