Analytics on Big Data

Riccardo Torlone
Università Roma Tre

ROMA TRE
UNIVERSITÀ DEGLI STUDI

Credits: Mohamed Eltabakh (WPI)

Analytics

- The discovery and communication of meaningful patterns in data (Wikipedia)
- It relies on **data analysis**
  - the process of cleaning, transforming, inspecting, and modeling data with the goal of discovering useful information
- Tools for data analysis
  - Data management
  - Business intelligence
  - Data mining (step of Knowledge Discovery)
  - Machine learning
  - Artificial intelligence

Why AI Would Be Nothing Without Big Data

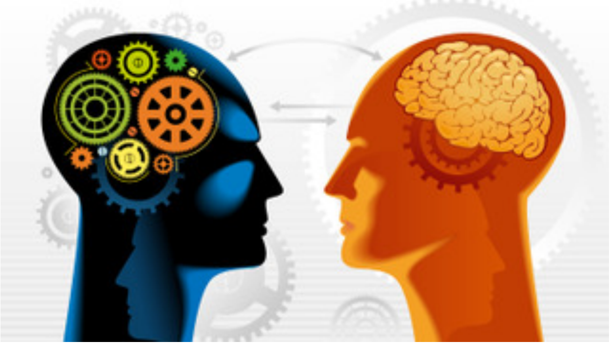
Bernard Marr, CONTRIBUTOR
FULL BIO
Opinions expressed by Forbes Contributors are their own.

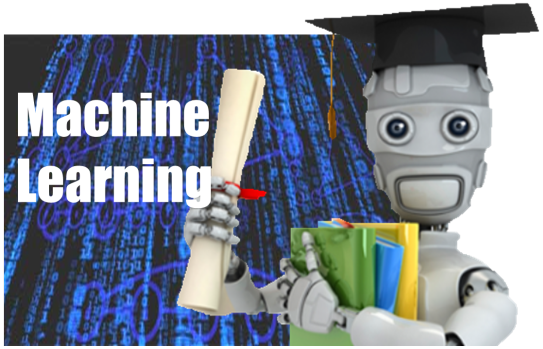
Artificial Intelligence (AI) is one of the most transformative forces of our times. While there may be debate whether AI will transform our world in good or evil ways, something we can all agree on is that AI would be nothing without big data.

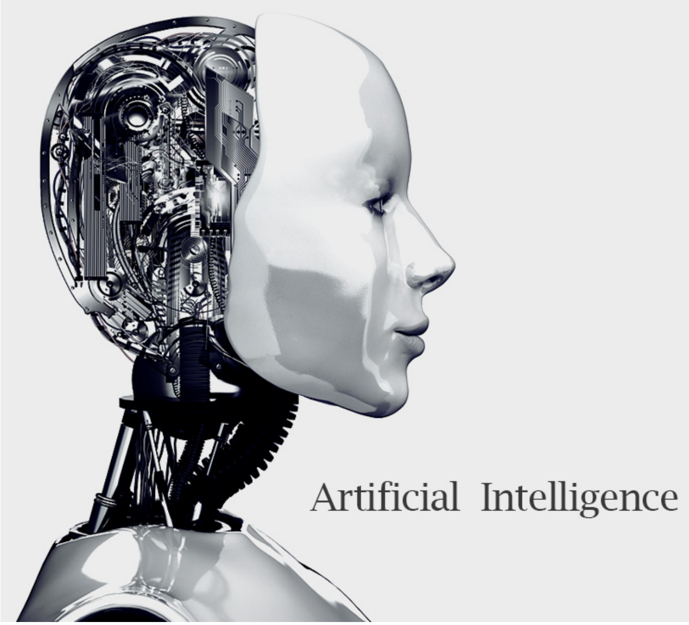
Even though AI technologies have existed for several decades, it's the explosion of data—the raw material of AI—that has allowed it to advance at incredible speeds. It's the billions of searches done every day on Google that provide a sizable real-time data set for Google to learn from our typos and search preferences. Siri and Cortana would have only a rudimentary understanding of our requests without the billions of hours of spoken word now digitally available that helped them learn our language. Similarly, Connie, the first concierge robot from Hilton Hotels understands natural language and responds to guests' questions about the hotel, local attractions, restaurants and more. The robot became intelligent due to the extensive data it was given to learn how to process future input.

Data analysis

- Analyze/mine/summarize large datasets
- Extract knowledge from past data
- Predict trends in future data
- Involve AI (machine learning) tools







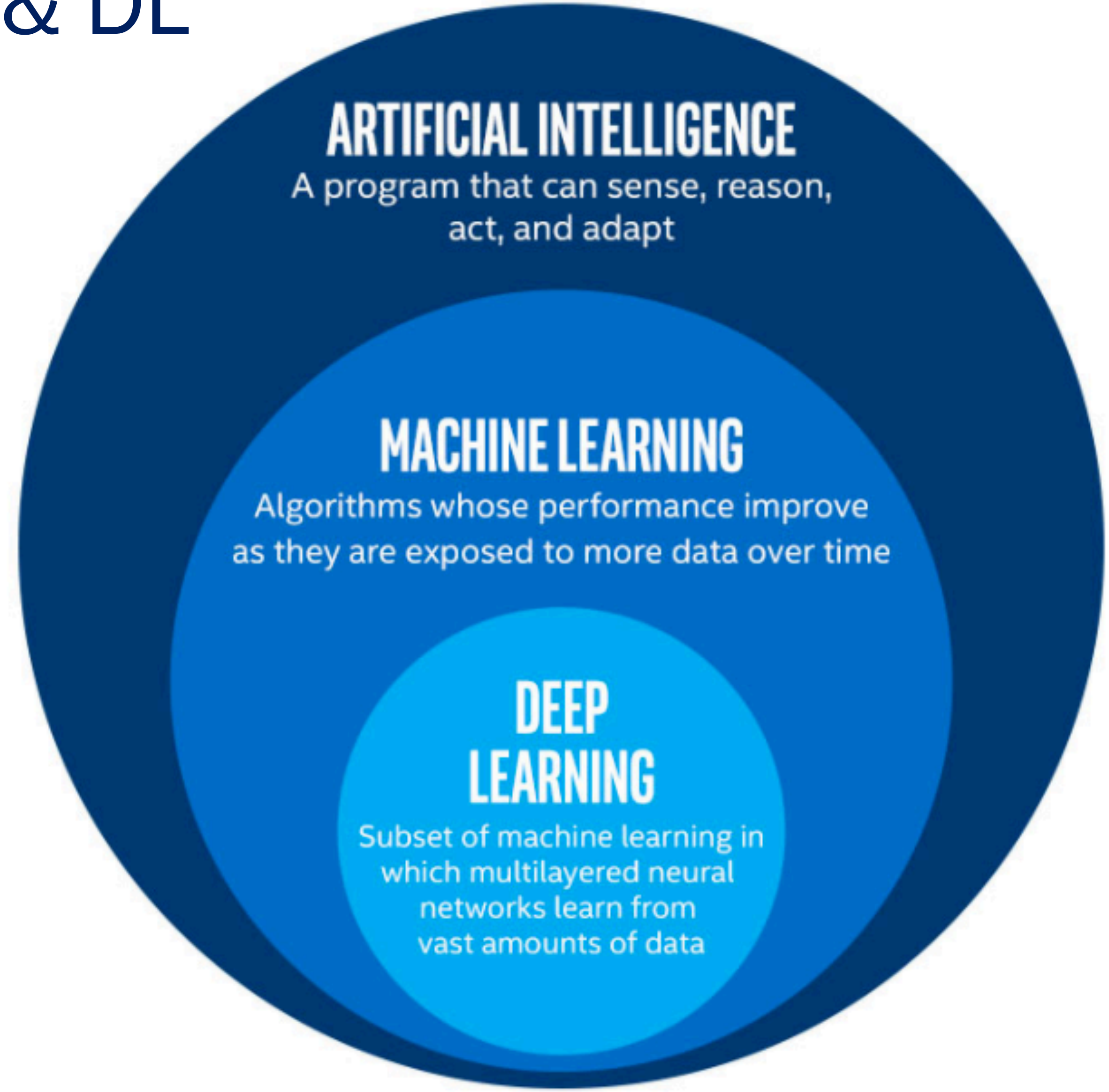
Common Use Cases

- Recommend products/friends/dates
- Classify content into predefined groups
- Group similar content together
- Find associations/patterns in actions/behaviors
- Detect anomalies/outliers
- Identify key topics/summarize text
- Ranking search results
- Others..

Examples of application

Retail/Marketing Identifying buying pattens of customers Finding associations among customer demographic characteristics Predicting response to mailing campaigns Market basket analysis
Banking Detecting patterns of fraudulent credit and use Identifying loyal customers Predicting customers likely to change their credit card affiliation Determining credit card spending by customer group
Insurance Claim analysis Predicting which customers will by new polices
Medicine characterizing patient behavior to predict surgery visits Identifying successful medical therapies for different illnesses

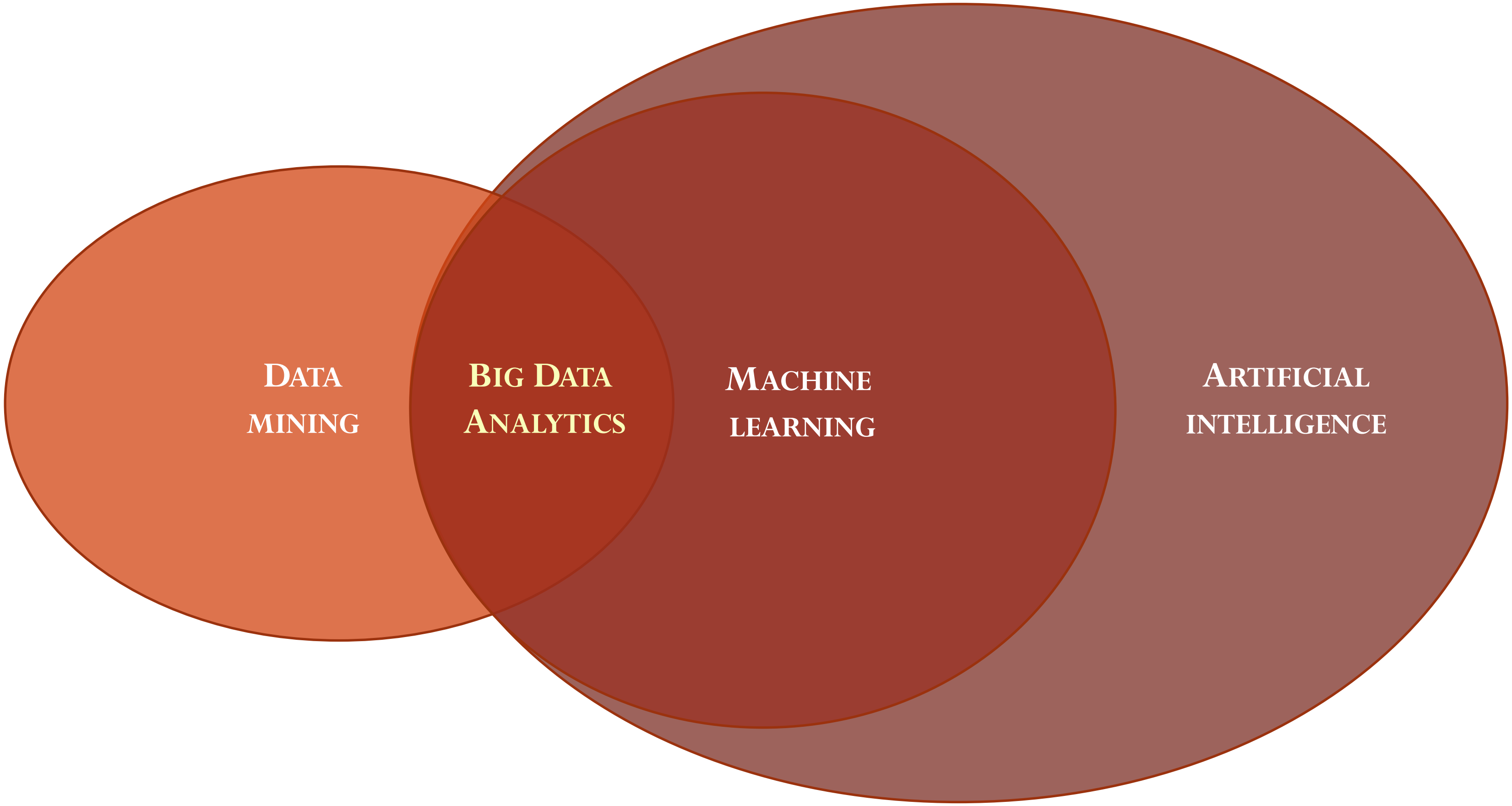
AI, ML & DL



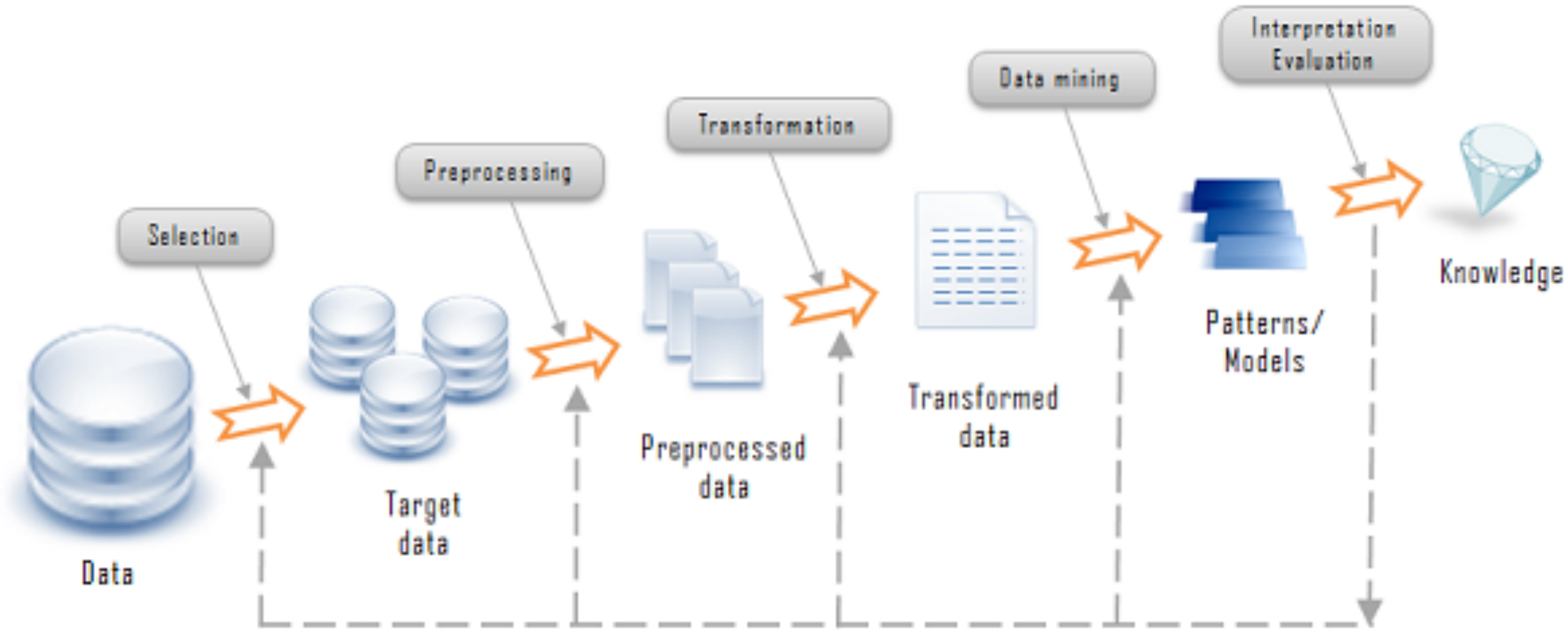
ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Data mining & machine learning

- Data analysis relies on datastore systems + ML/DM
- Machine learning:
  - branch of artificial intelligence
  - capability to learn from data without being explicitly programmed
  - example: distinguish between spam and non-spam messages
- Data mining:
  - step of the "Knowledge Discovery" process
  - discovering patterns in large data sets for decision support
  - example: regularities between products sold in large transaction data
- Data mining involves machine learning techniques

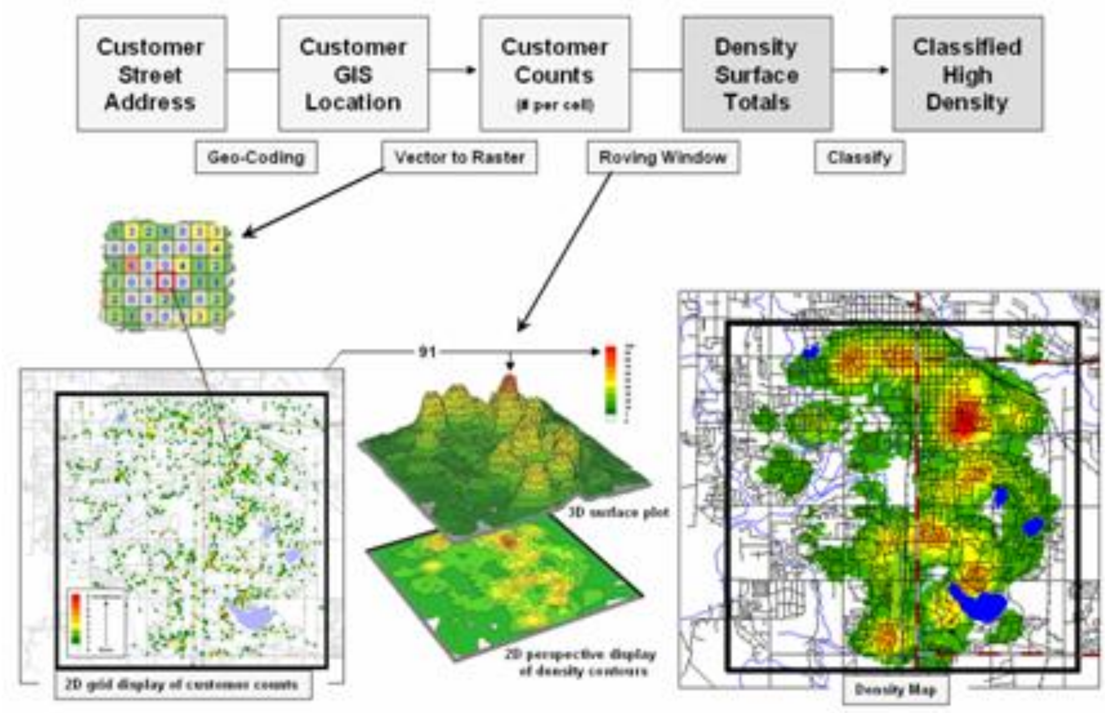

Data mining
Big Data Analytics
Machine learning
Artificial intelligence

The knowledge discovery process



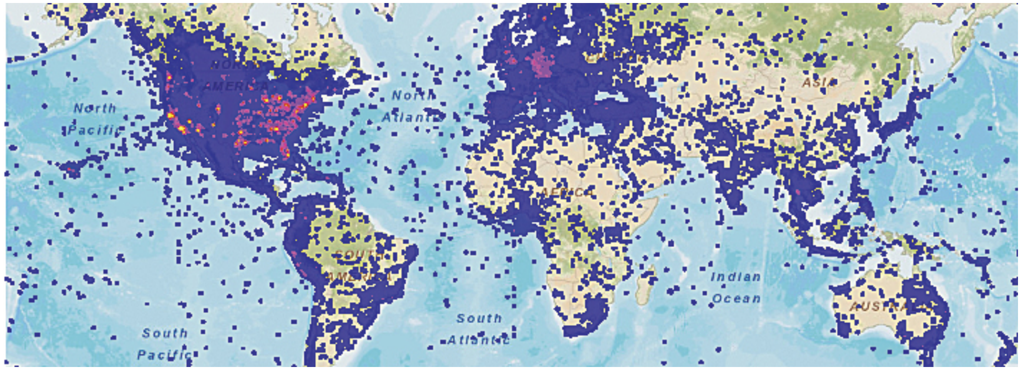
Data mining

- Discovering **patterns** in large data sets and transform them into an understandable structure for further use
- Methods from:
  - database management,
  - statistics,
  - machine learning,
  - visualization

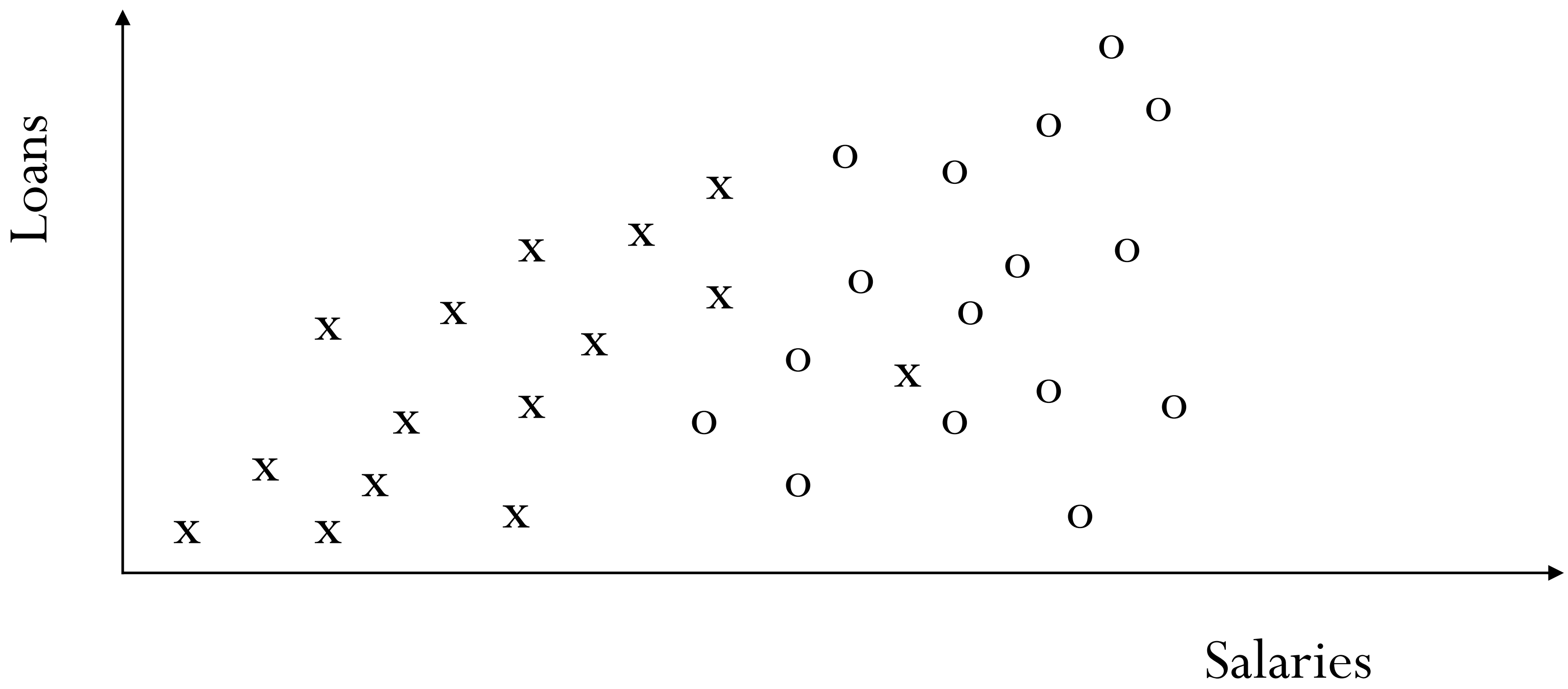


Data & Patterns

- Input: raw data
  - Big data
  - Usually unstructured
  - Coming from different sources
- Output: patterns
  - Expression, in an appropriate language, that describes succinctly some information extracted from the data
  - Features
    - Regularities on data
    - High-level information



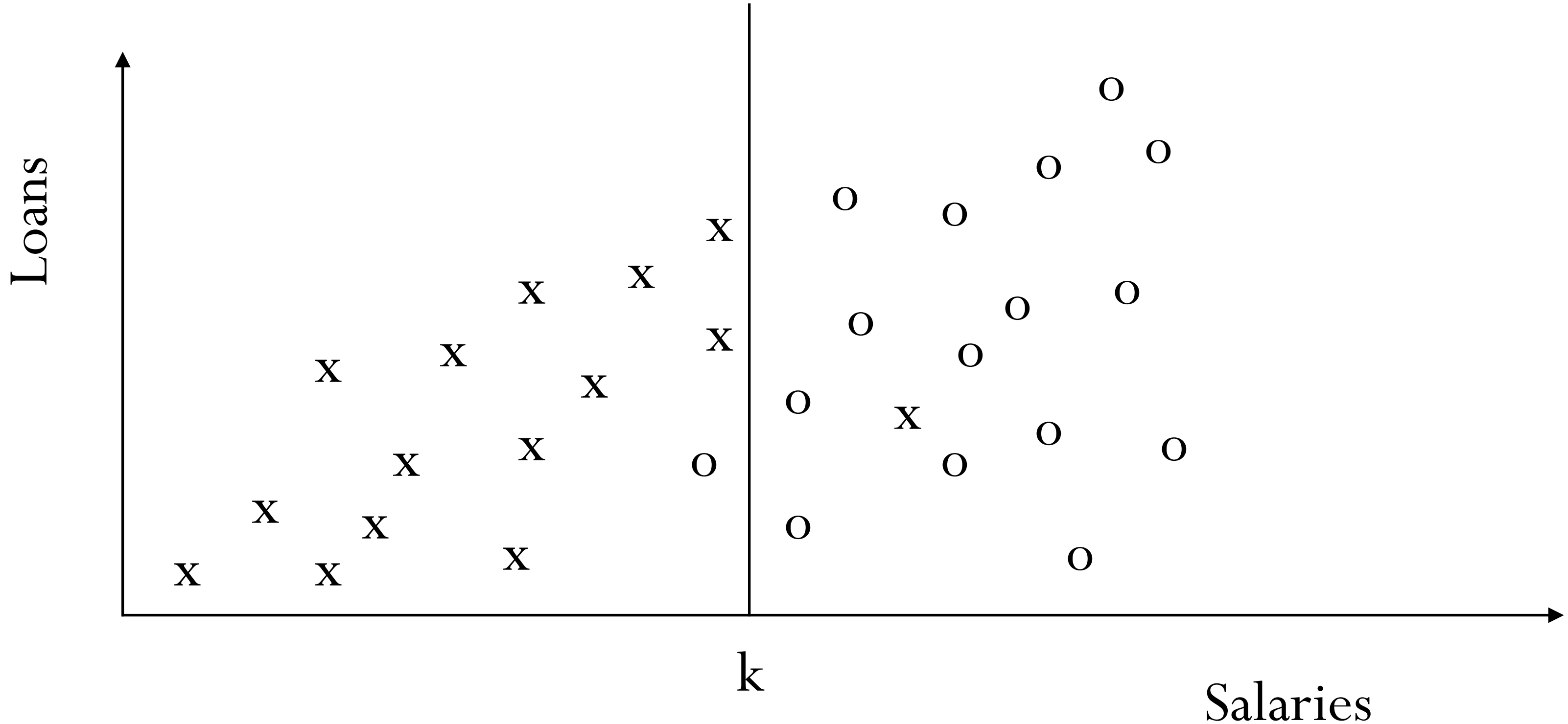
Example



Bank loans

- x: missed rate payments
- o: regular payments

Example

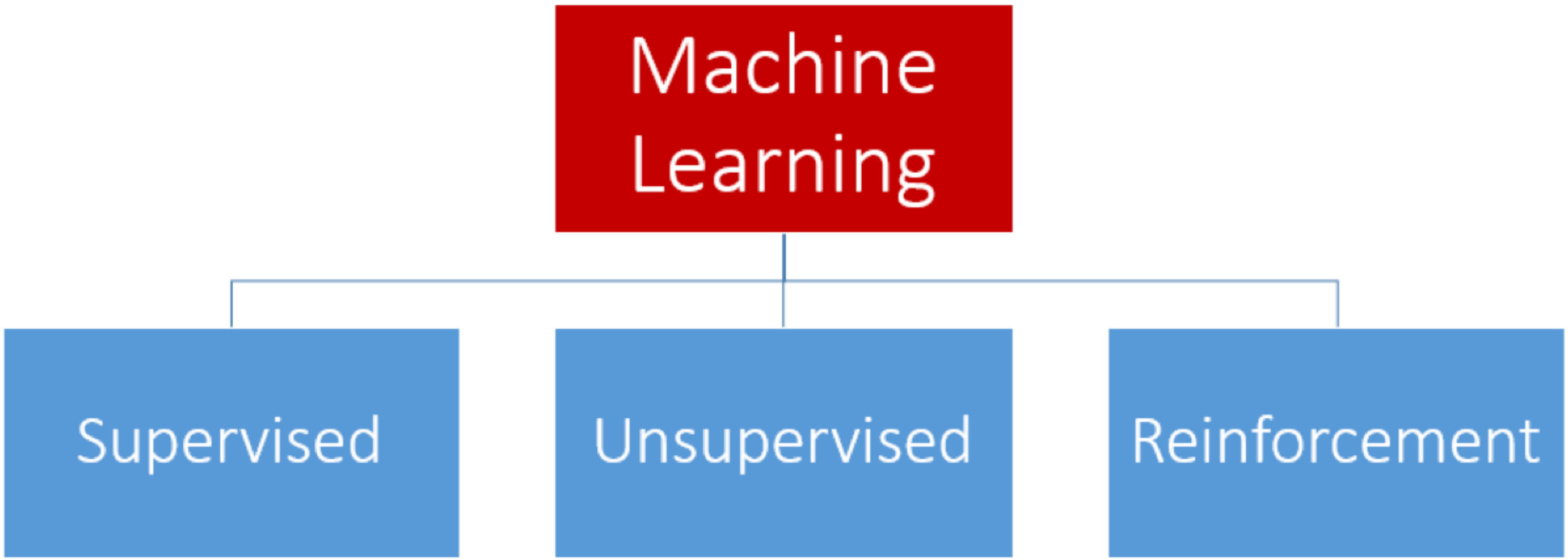


- Pattern: IF salary < k THEN missed payments

Properties of patterns

- Validity
  - within some degree (shifting k to the right reduces validity)
- Novelty
  - with respect to previous knowledge
- Utility
  - for example: increase in expected profit from a bank
- Comprehensibility
  - Syntactical measures
  - Semantic measures

A possible classification



Task driven (Regression / Classification)

Data driven (Clustering)

Algorithm learns to react to an environment

Tools

- Apache Mahout
- MLlib
- Weka
- KNIME
- mlpy
- OpenNN
- dlib
- Orange
- Scikit-learn
- R
- RapidMiner
- Shogun
- …

In Our Context…

- Efficient in analyzing/mining data
- Do not scale

- Efficient in managing big data
- Not so easy to analyze or mine the data

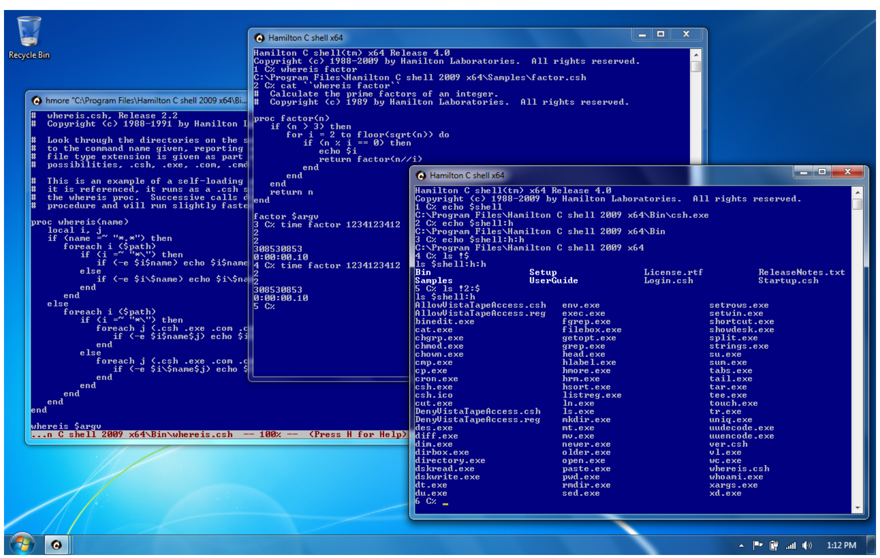
How to integrate these two worlds together

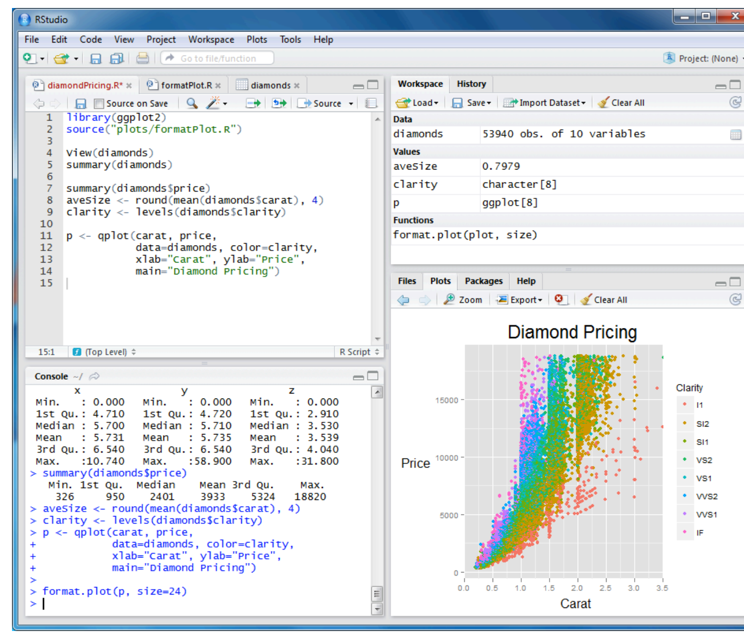
Big data projects

- R over a cluster computing framework
  - Rhadoop/RHive:
    - Open source extension of R on Hadoop
  - Revolution R:
    - R distribution from Microsoft
  - SparkR (R on Spark)
  - Many other connectors
- Apache Mahout
  - Open-source package on Hadoop for data mining and machine learning
- Apache MLlib
  - Spark's scalable machine learning library consisting of common learning algorithms and utilities

R

- A software environment for statistical computing and graphics
- Widely used among statisticians
- Open source (GNU project)
- Binary versions for various operating systems
- Uses a command line interface but GUIs are also available





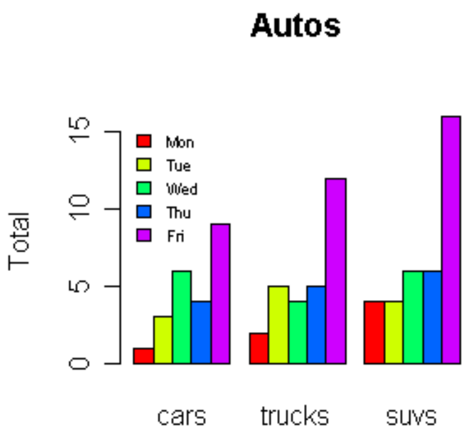
Main features of R

- Data structures:
  - vectors, matrices, arrays, data frames (tables) and lists
  - a scalar is represented as a vector with length one in R.
- Supports:
  - procedural programming with tons of built-in functions
  - a wide variety of statistical techniques
    - linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …
  - matrix arithmetic
  - several graphical facilities
  - highly extensible
  - object-oriented programming with generic functions

Examples in R

```
# Function that return multiple objects
powers <- function(x) {
  parcel = list(x2=x*x, x3=x*x*x, x4=x*x*x*x);
  return(parcel);
}
X = powers(3);
print("Showing powers of 3 --"); print(X);

# Read values from tab-delimited autos.dat
autos_data <- read.table("C:/R/autos.dat", header=T, sep="\t")
# Graph autos with adjacent bars using rainbow colors
barplot(as.matrix(autos_data), main="Autos", ylab= "Total",
   beside=TRUE, col=rainbow(5))
# Place the legend at the top-left corner with no frame
# using rainbow colors
legend("topleft", c("Mon","Tue","Wed","Thu","Fri"), cex=0.6,
   bty="n", fill=rainbow(5));
```

cars	trucks	suvs
1	2	4
3	5	4
6	4	6
4	5	6
9	12	16



Apache Mahout

- Apache Software Foundation project
- Goals:
  - Build a scalable machine learning library.
    - On large data sets.
    - On a community of developers
  - Be as fast and efficient given the intrinsic design of the algorithm
    - Core algorithms implemented on top of scalable, distributed systems
    - Mahout implementations on distributed environments
    - Solutions that run on a single machine are also provided
  - Be open source.

Components of Apache Mahout

- An environment for building scalable algorithms
- "standard" implementations
  - A library of ML algorithm implementations
  - It runs over Hadoop with MapReduce/Spark/H2O/Flink engines
- Samsara
  - A library of ML algorithm implementations using linear algebra and statistical operations along with the data structures to support them
  - Scala with specific extensions that look like R
  - It runs over Hadoop with the Spark engine

Machine learning algorithms in Mahout

- 3C:
  - Collaborative Filtering
  - Clustering
  - Classification
- FPM:
  - Frequent Pattern Mining
- Miscellaneous:
  - Dimensionality Reduction (for features extraction)
  - Topic Models (for topics discovering)
  - Others

Apache MLlib

- MLlib is Apache Spark's scalable machine learning library
- Usable in Java, Scala, Python, and R.
- MLlib fits into Spark's APIs and interoperates with NumPy in Python (as of Spark 0.9) and R libraries (as of Spark 1.5).
- You can use any Hadoop data source (e.g. HDFS, HBase, or local files), making it easy to plug into Hadoop workflows.
- If you have a Hadoop 2 cluster, you can run Spark and MLlib without any pre-installation.

Machine learning algorithms in MLlib

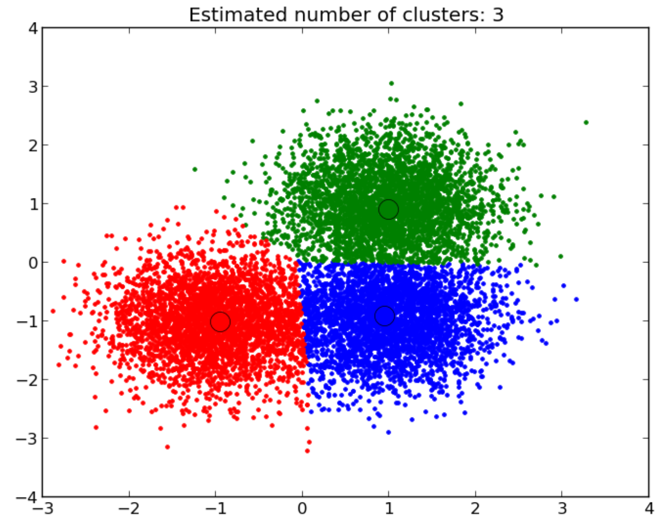
- 3C:
  - Classification: logistic regression, naive Bayes,…
  - Clustering: K-means, Gaussian mixtures (GMMs),…
  - Collaborative Filtering: alternating least squares (ALS)
- FPM:
  - Frequent itemsets, association rules, and sequential pattern mining
- Miscellaneous:
  - Regression: generalized linear regression, survival regression,…
  - Decision trees, random forests, and gradient-boosted trees
  - Topic modeling: latent Dirichlet allocation (LDA)

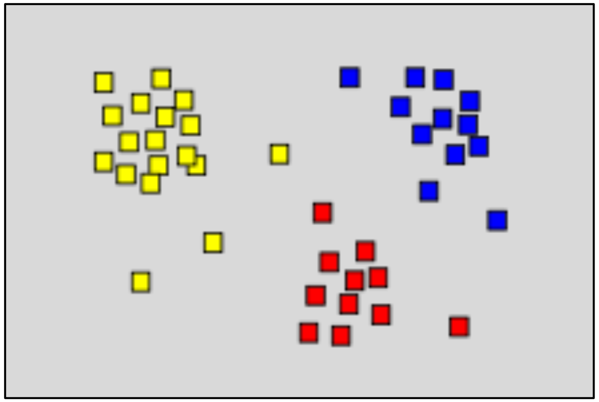
C1: Collaborative Filtering

- A set of techniques for automatic recommendation
- Goal:
  - Predict the interest of a user for an item
  - Filter only interesting items
- "Collaborative" approach:
  - Collecting preferences information from many users
- Rationale:
  - if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly.
- Requirement:
  - Collect a large number of user preferences

C2: Clustering

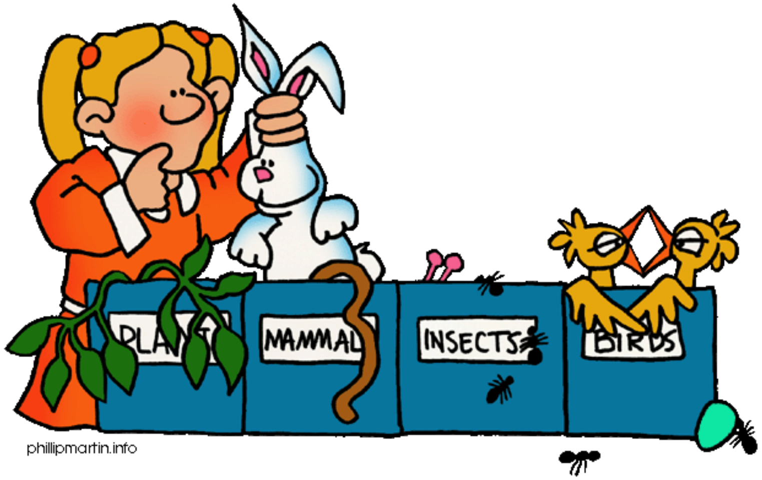
- Group similar objects together
- Different distance measures
  - Manhattan, Euclidean, …
- Different algorithms:
  - Connectivity based,
  - Centroid-based,
  - Distribution-based,
  - Density-Based,
  - Others





C3: Classification

- Place items into predefined categories:
  - Entertainment, politics, risks, ..
  - Recommenders
- Approaches:
  - Linear,
  - Decision trees,
  - Bayesian networks,
  - Support vector machines,
  - Neural networks,
  - …



FPM:

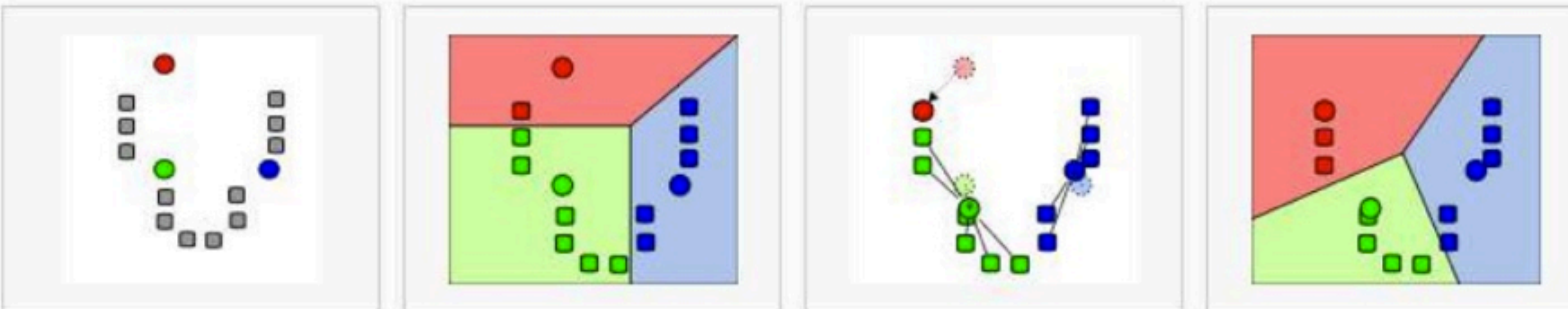
- Find the frequent itemsets
  - milk, bread, cheese are sold frequently together
- Very common in
  - market analysis,
  - access pattern analysis,
  - …

We Focus On...

- Clustering
  - K-Means
- Classification
  - Naïve Bayes
- Frequent Pattern Mining
  - Apriori

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 converges

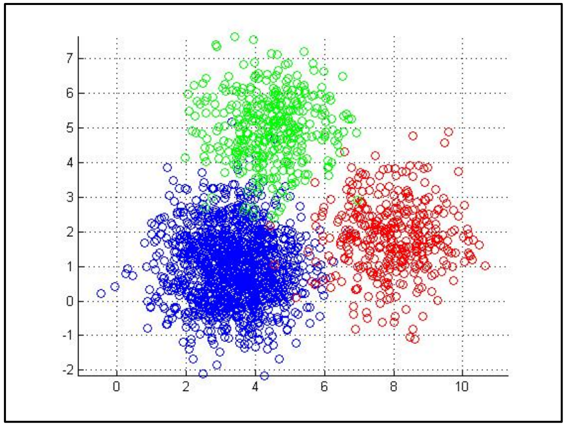
K-Means Algorithm

- Step 1: **Select** K points at random (Centers)
- Step 2: **For each** data point, assign it to the closest center
  - Now we formed K clusters
- Step 3: **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
- Step 4: **If** the new centers are different from the old centers (previous iteration) **then** Go to Step 2

K-Means in MapReduce

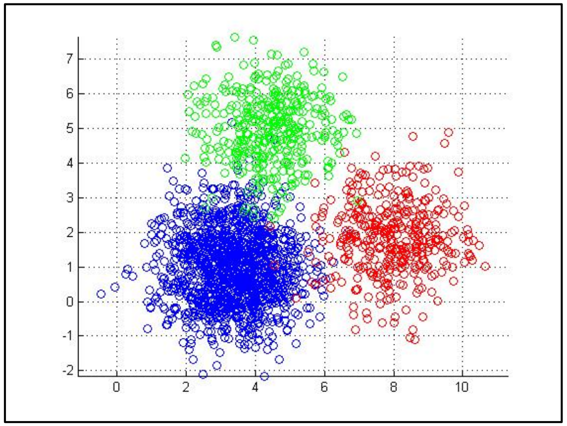
- Input
  - Dataset (set of points in 2D) – Large
  - Initial centroids (K points) – Small

- Map step
  - Each map reads the K-centroids + one block from dataset
  - Assign each point to the closest centroid
  - Output <centroid, point>



K-Means in MapReduce (Cont'd)

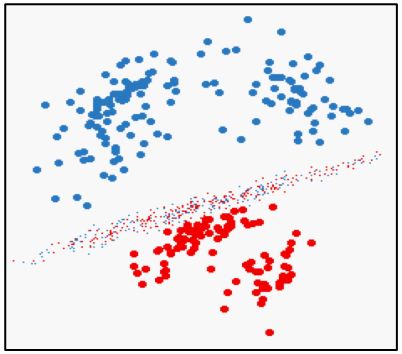
- Reduce step
  - Gets all points for a given centroid
  - Re-compute a new centroid for this cluster
  - Output: <new centroid>



- Iteration Control
  - Compare the old and new set of K-centroids
    - **if** similar **then** Stop
    - **else**
      - **if** max iterations has reached
        - **then** Stop
        - **else** Start another Map-Reduce Iteration

Naïve Bayes Classifier

- Given a dataset (training data), we learn (build) a statistical model
  - This model is called “Classifier”
- Each point in the training data is in the form of:
  - <label, feature_1, feature_2,…,feature_N>
  - Label: is the class label
  - Features 1..N: the features (dimensions of the point)



- Then, given a point without a label <??, feature1, …,featureN>
  - Use the model to decide on its label

Naïve Bayes Classifier: Example

Three features

Class label (male or female)

sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Training dataset

Naïve Bayes Classifier (Cont'd)

- For each feature in each label
  - Compute the mean and variance

sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

That is the model (classifier)

Naïve Bayes: Classify New Object

Male or female?

sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

- For each label: Compute posterior value
- The label with the largest posterior is the suggested label

$$posterior(male) = \frac{P(male)\,p(height|male)\,p(weight|male)\,p(footsize|male)}{evidence}$$

$$posterior(female) = \frac{P(female)\,p(height|female)\,p(weight|female)\,p(footsize|female)}{evidence}$$

Naïve Bayes: Classify New Object (Cont'd)

Male or female? →

sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

$$posterior(male) = \frac{P(male)\, p(height|male)\, p(weight|male)\, p(footsize|male)}{evidence}$$

$$posterior(female) = \frac{P(female)\, p(height|female)\, p(weight|female)\, p(footsize|female)}{evidence}$$

- Evidence: Can be ignored since it is the same constant for all labels
- P(label): % of training points with this label
- p(feature | label) = $\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(f-\mu)^2}{2\sigma^2}\right)$, f is the feature value in sample

Frequent Pattern Mining

- **Input**:
  - a set of items I = {milk, bread, jelly, …}
  - a set of transactions where each transaction contains subset of items
    - t1 = {milk, bread, water}
    - t2 = {milk, nuts, butter, rice}
- **Goal:**
  - What are the itemsets frequently sold together ?
  - Is there a relationship between itemsets in transactions?

Output: association rules

- Rules $X \Rightarrow Y$ where X,Y appears together in a transaction
- **Support S**: #trans. containing $X \cup Y$ / #trans. in D
  - Statistical relevance
- **Confidence C**: #trans. containing $X \cup Y$ / #trans. containing X
  - Relevance of the implication

Example

$$\text{Milk} \Rightarrow \text{Eggs}$$

- Support:
  - 2% of transactions contain both elements
- Confidence:
  - 30% of transactions that contain milk contain eggs as well

Example

TRANSACTION ID	TRANSACTIONS
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- Let us assume:
  - Minimal support: 50%
  - Minimal confidence:50%

Examples (cont)

TRANSACTION ID	TRANSACTIONS
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- Extracted rules:
  - A $\Rightarrow$ C support 50% confidence 66.6
  - C $\Rightarrow$ A support 50% confidence 100%

Approach (cont)

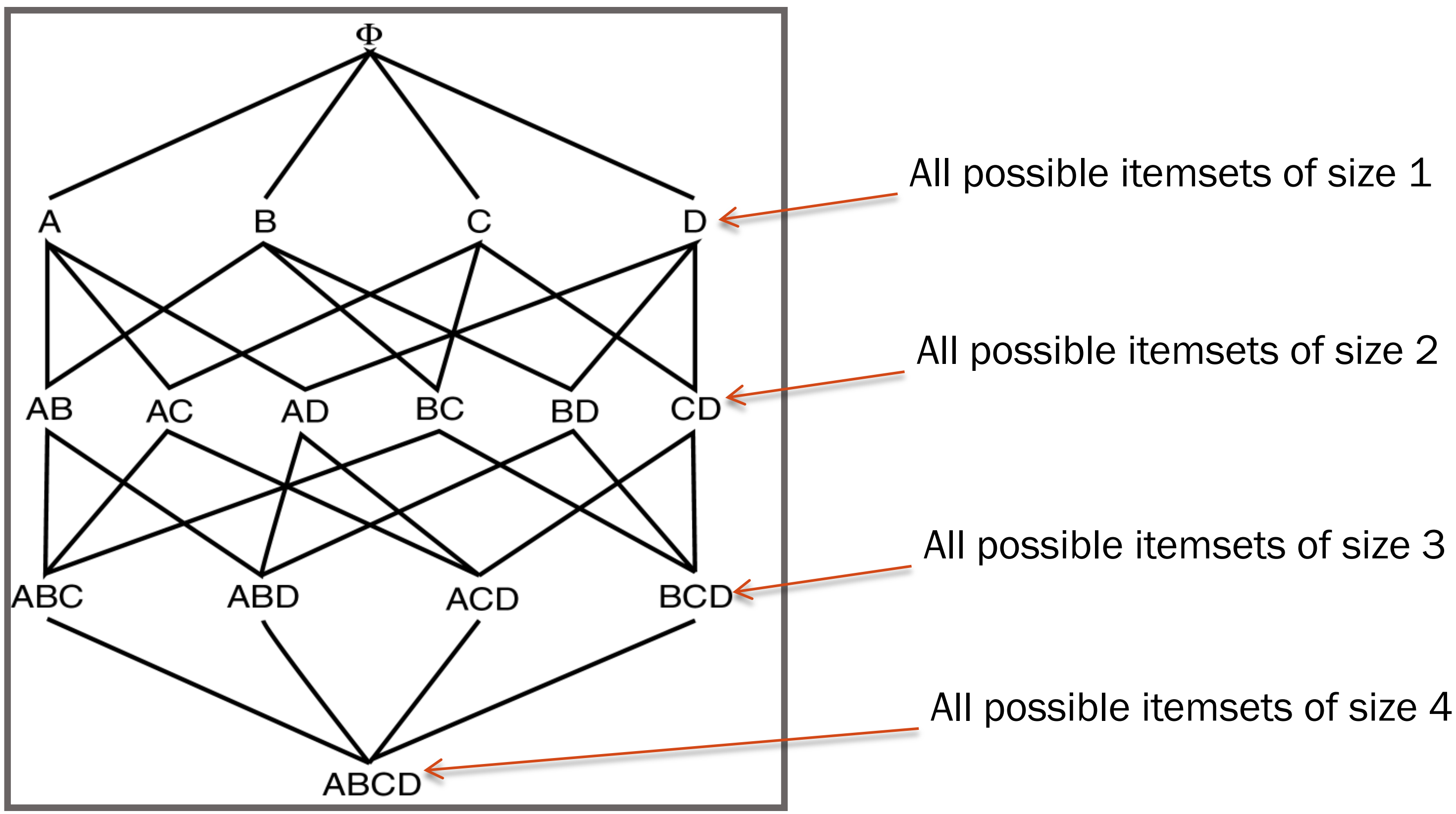
- Step 2: rule extraction
  - Minimal confidence 50%
  - Confidence of rule A $\Rightarrow$ C
    - Support {A,C} / Support {A} = 66.6%
  - Confidence of rule C $\Rightarrow$ A
    - Support {A,C} / Support {C} = 100%
  - Extracted rules
    - A $\Rightarrow$ C support 50%, confidence 66.6%
    - C $\Rightarrow$ A support 50%, confidence 100%

Application

- Market basket analysis
  - $* \Rightarrow$ eggs
    - What should we promote to increase the sales of eggs?
  - Milk $\Rightarrow *$
    - what products need to be sold by a supermarket that sells milk?

How to find frequent itemsets

- Naïve Approach
  - Enumerate all possible itemsets and then count each one



Can we optimize??

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

- {Bread} : 80%
- {PeanutButter} : 60%
- {Bread, PeanutButter} : 60%
- Important property:
  - For itemset S={X,Y, Z, …} of size n to be frequent, all its subsets of size n-1 must be frequent as well

Apriori Algorithm

- Executes in scans (iterations), each scan has two phases
  - Given a list of candidate itemsets of size n, count their appearance and find frequent ones
  - From the frequent ones generate candidates of size n+1 (previous property must hold)
- Start the algorithm where n =1, then repeat

Apriori Example

Transaction	Items
t_1	**Blouse**
t_2	**Shoes,Skirt,TShirt**
t_3	**Jeans,TShirt**
t_4	**Jeans,Shoes,TShirt**
t_5	**Jeans,Shorts**
t_6	**Shoes,TShirt**
t_7	**Jeans,Skirt**
t_8	**Jeans,Shoes,Shorts,TShirt**
t_9	**Jeans**
t_{10}	**Jeans,Shoes,TShirt**
t_{11}	**TShirt**
t_{12}	**Blouse,Jeans,Shoes,Skirt,TShirt**
t_{13}	**Jeans,Shoes,Shorts,TShirt**
t_{14}	**Shoes,Skirt,TShirt**
t_{15}	**Jeans,TShirt**
t_{16}	**Skirt,TShirt**
t_{17}	**Blouse,Jeans,Skirt**
t_{18}	**Jeans,Shoes,Shorts,TShirt**
t_{19}	**Jeans**
t_{20}	**Jeans,Shoes,Shorts,TShirt**

Apriori Example (Cont'd)

Scan	Candidates	Large Itemsets
1	{Blouse},{Jeans},{Shoes}, {Shorts},{Skirt},{TShirt}	{Jeans},{Shoes},{Shorts} {Skirt},{Tshirt}
2	{Jeans,Shoes},{Jeans,Shorts},{Jeans,Skirt}, {Jeans,TShirt},{Shoes,Shorts},{Shoes,Skirt}, {Shoes,TShirt},{Shorts,Skirt},{Shorts,TShirt}, {Skirt,TShirt}	{Jeans,Shoes},{Jeans,Shorts}, {Jeans,TShirt},{Shoes,Shorts}, {Shoes,TShirt},{Shorts,TShirt}, {Skirt,TShirt}
3	{Jeans,Shoes,Shorts},{Jeans,Shoes,TShirt}, {Jeans,Shorts,TShirt},{Jeans,Skirt,TShirt}, {Shoes,Shorts,TShirt},{Shoes,Skirt,TShirt}, {Shorts,Skirt,TShirt}	{Jeans,Shoes,Shorts}, {Jeans,Shoes,TShirt}, {Jeans,Shorts,TShirt}, {Shoes,Shorts,TShirt}
4	{Jeans,Shoes,Shorts,TShirt}	{Jeans,Shoes,Shorts,TShirt}
5	$\emptyset$	$\emptyset$

Machine learning over Hadoop

- How to implement K-Means, Naïve Bayes and FMP in MapReduce?
- And in Spark??

Apache Mahout

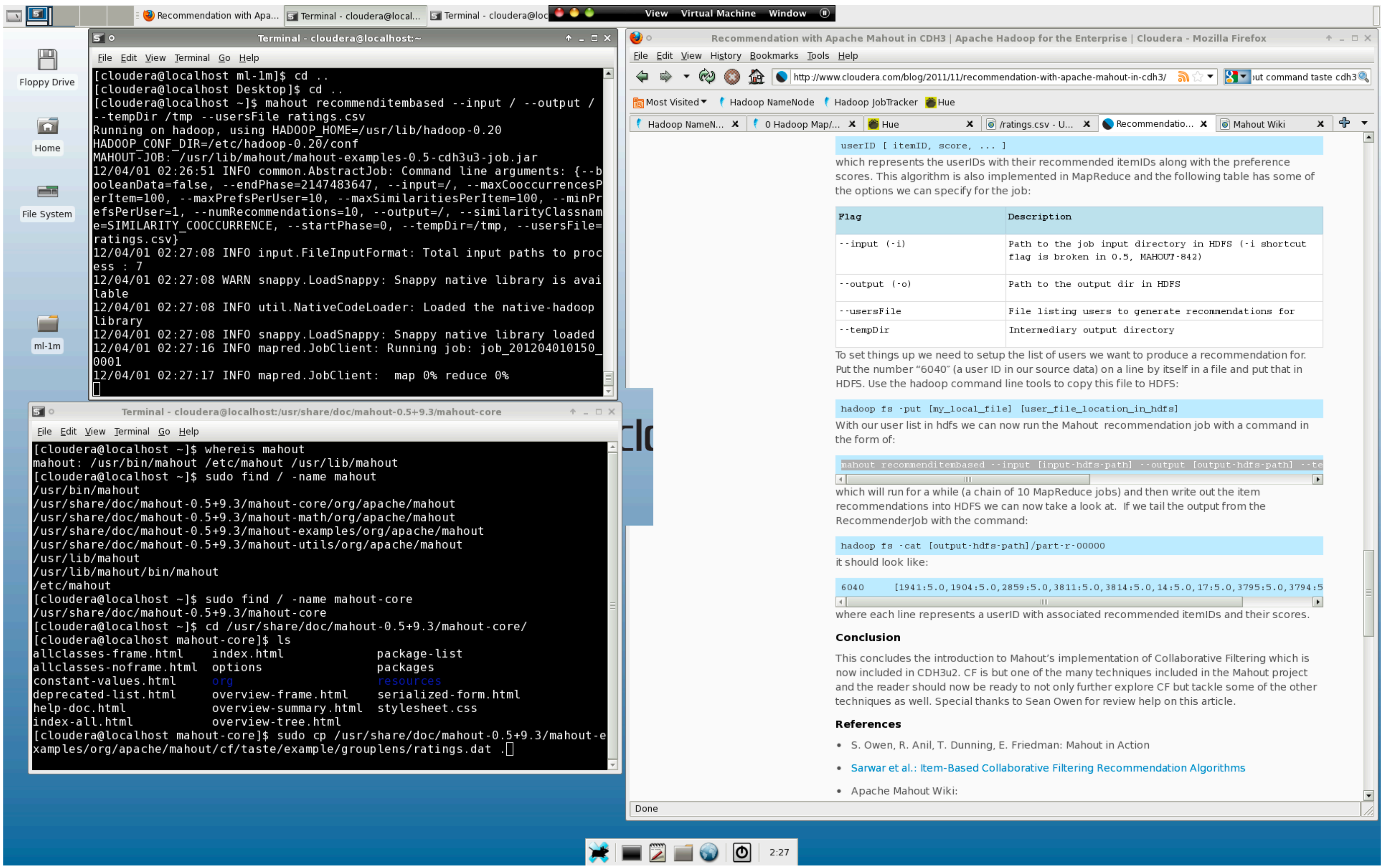
- http://mahout.apache.org/

```
bin/mahout kmeans \
    -i <input vectors directory> \
    -c <input clusters directory> \
    -o <output working directory> \
    -k <optional number of initial clusters to sample from input vectors>
    -dm <DistanceMeasure> \
    -x <maximum number of iterations> \
    -cd <optional convergence delta. Default is 0.5> \
    -ow <overwrite output directory if present>
    -cl <run input vector clustering after computing Canopies>
    -xm <execution method: sequential or mapreduce>
```

Apache Mahout example

```
import org.apache.mahout.cf.taste.impl.model.file.*;
import org.apache.mahout.cf.taste.impl.neighborhood.*;
import org.apache.mahout.cf.taste.impl.recommender.*;
import org.apache.mahout.cf.taste.impl.similarity.*;
import org.apache.mahout.cf.taste.model.*;
import org.apache.mahout.cf.taste.neighborhood.*;
import org.apache.mahout.cf.taste.recommender.*;
import org.apache.mahout.cf.taste.similarity.*;

class RecommenderIntro {

  private RecommenderIntro() {
  }

  public static void main(String[] args) throws Exception {

    DataModel model = new FileDataModel(new File("intro.csv"));
    UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
    UserNeighborhood neighborhood = new NearestNUserNeighborhood(2, similarity, model);

    Recommender recommender = new GenericUserBasedRecommender(
        model, neighborhood, similarity);

    List<RecommendedItem> recommendations = recommender.recommend(1, 1);

    for (RecommendedItem recommendation : recommendations) {
      System.out.println(recommendation);
    }
  }
}
```

Apache Mahout at work



Apache MLlib example

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

# Load and parse the data
data = sc.textFile("data/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, initializationMode="random")

# 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, "target/org/apache/spark/PythonKMeansExample/KMeansModel")
sameModel = KMeansModel.load(sc, "target/org/apache/spark/PythonKMeansExample/KMeansModel")
```

Apache MLlib at work