

Analytics on Big Data

Riccardo Torlone
Università Roma Tre



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.

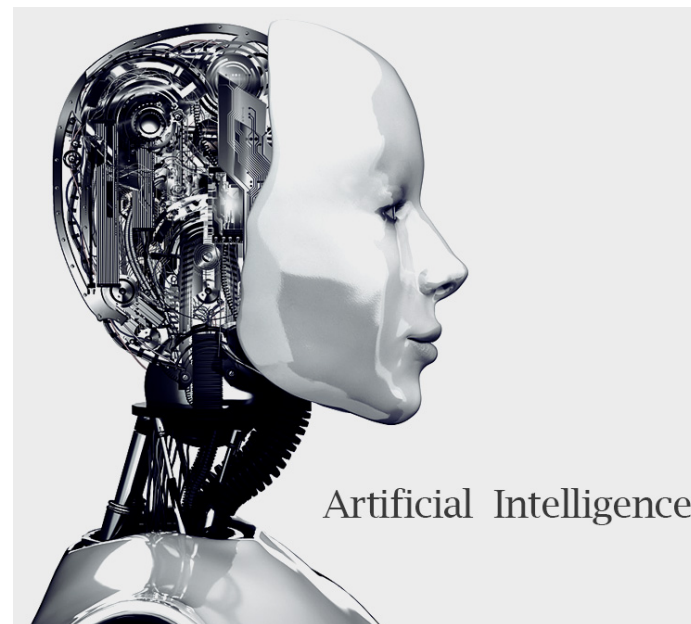
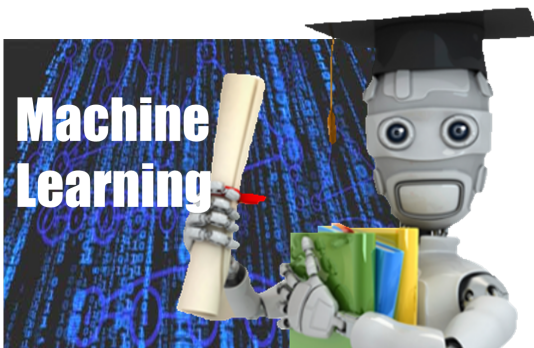
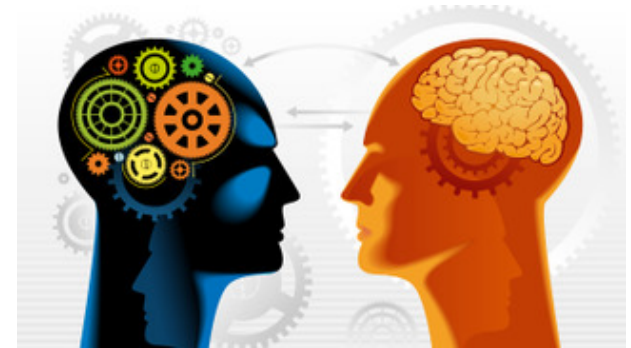
Artificial Intelligence (AI) is one of the most transformative forces of our times. While there may be debate whether AI will transform our world in good or evil ways, something we can all agree on is that AI would be nothing without big data.

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 patterns 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

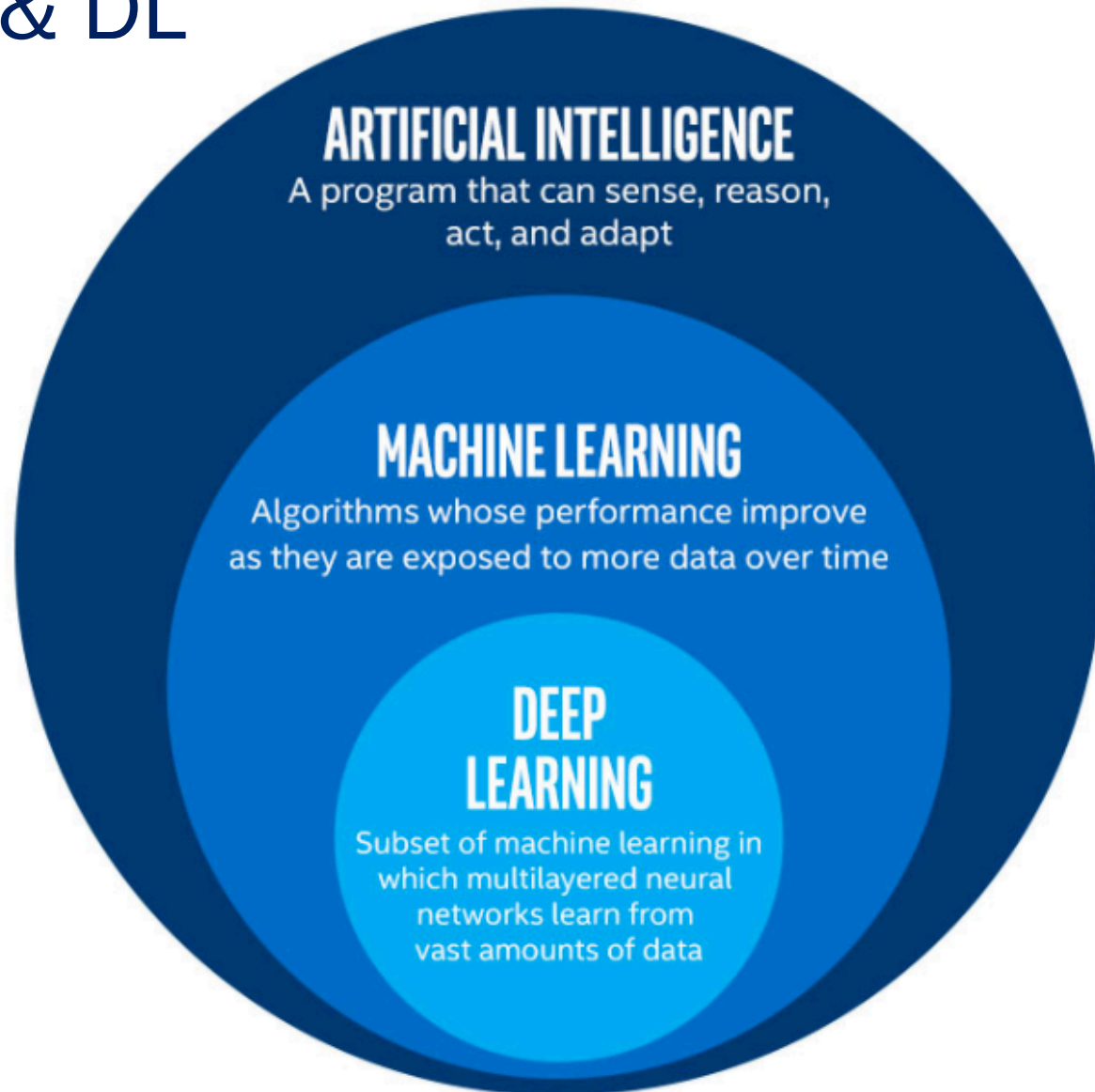
Predicting which customers will buy new policies

Medicine

characterizing patient behavior to predict surgery visits

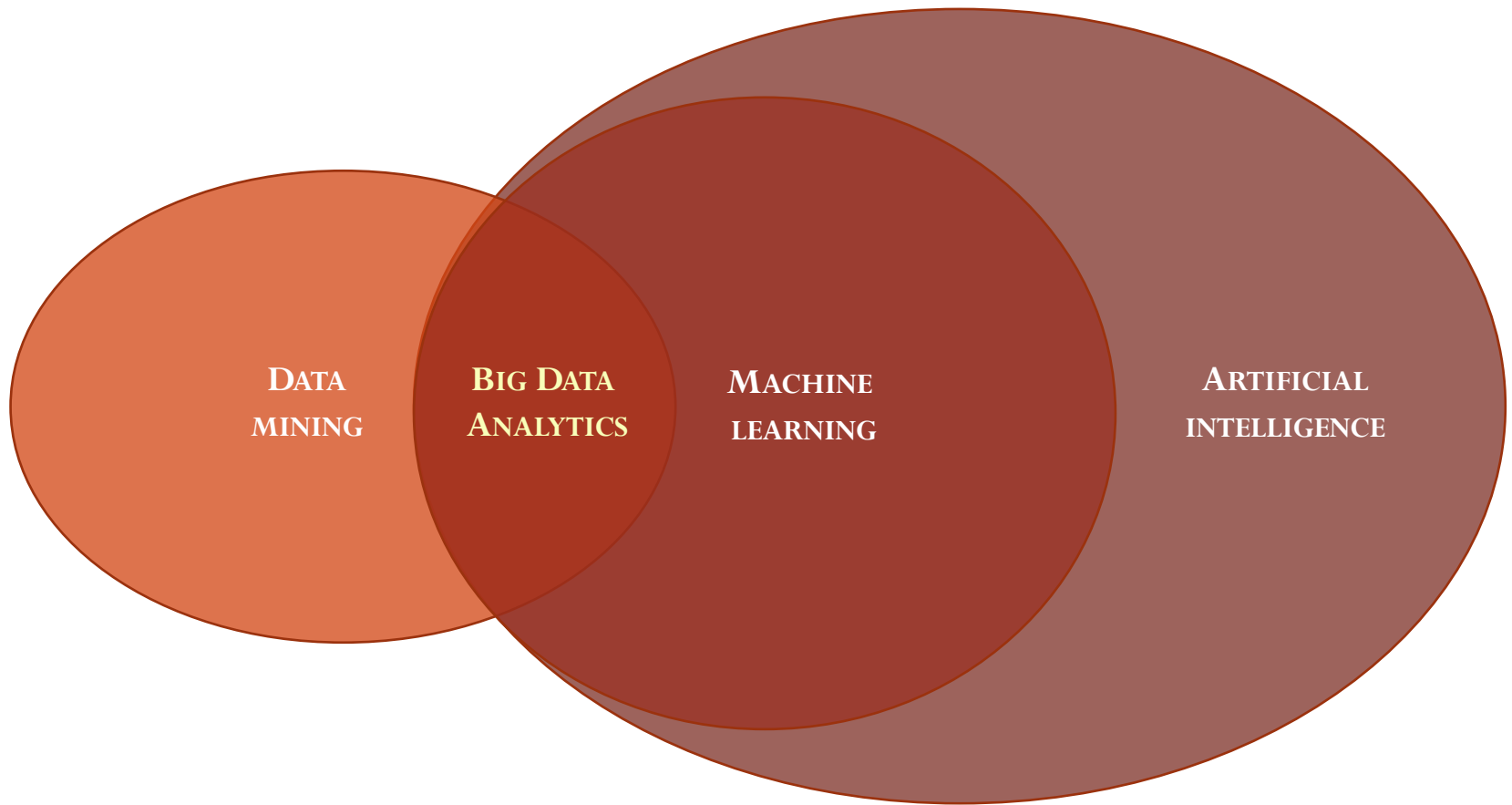
Identifying successful medical therapies for different illnesses

AI, ML & DL

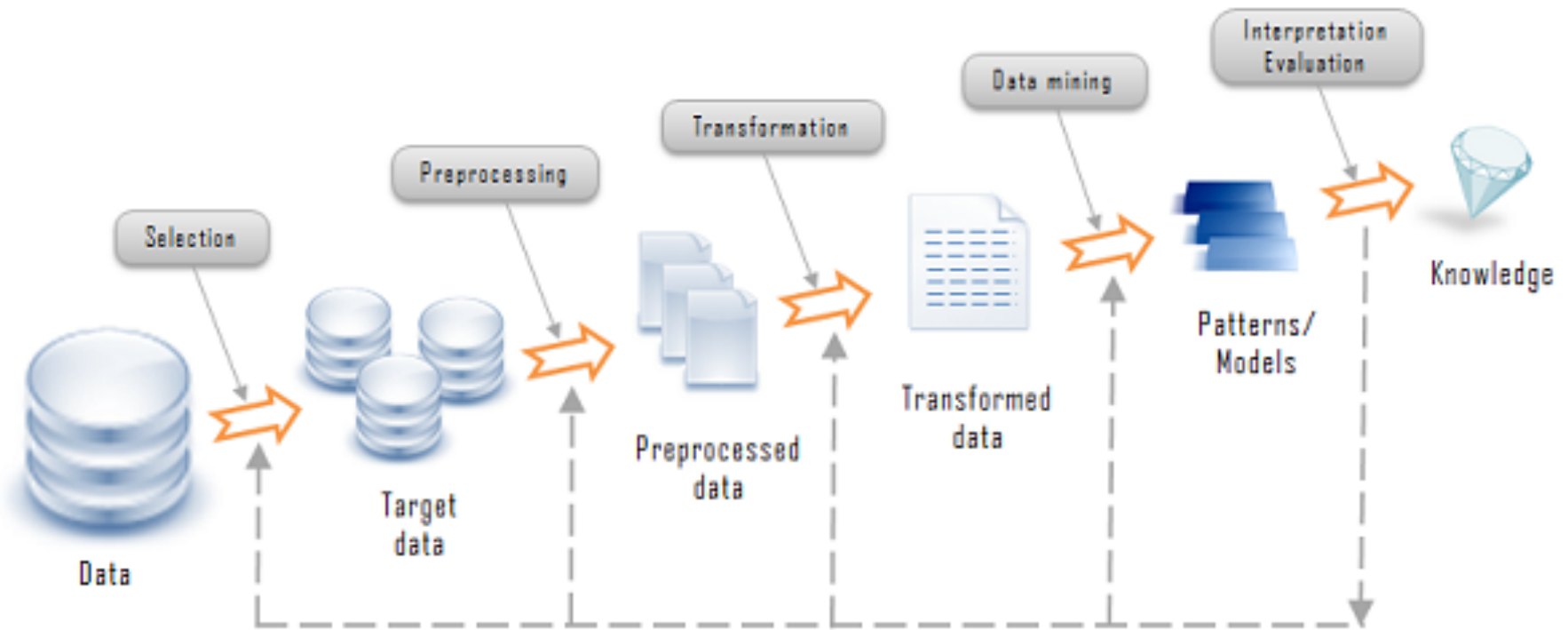


Data mining & machine learning

- 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

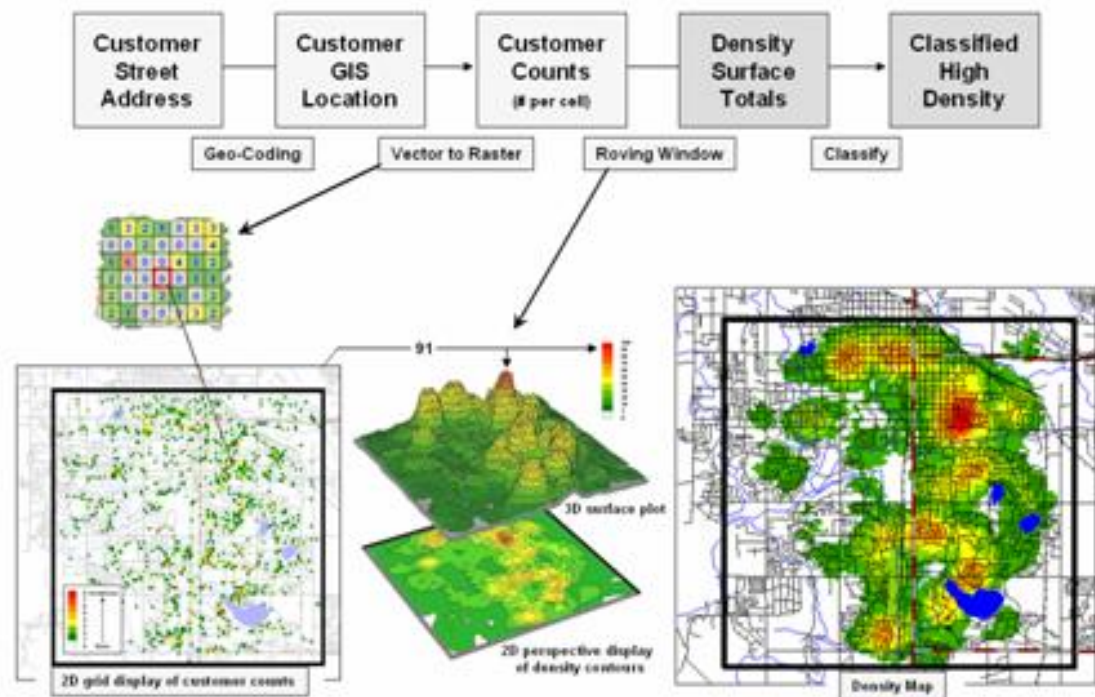


The knowledge discovery process



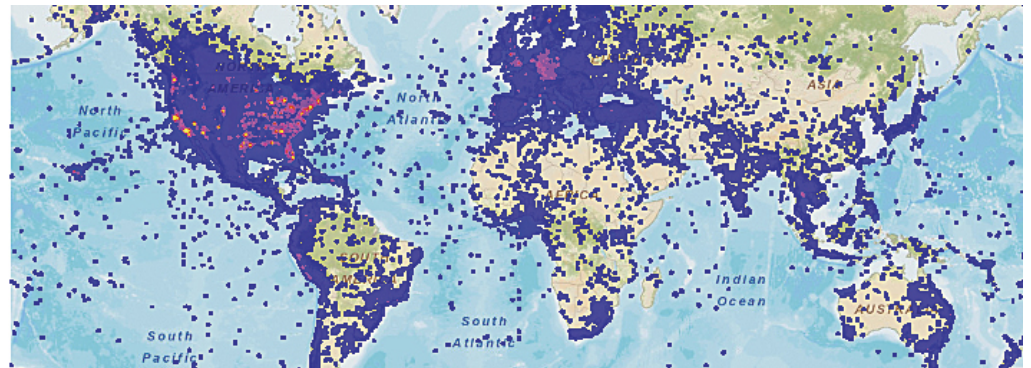
Data mining

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

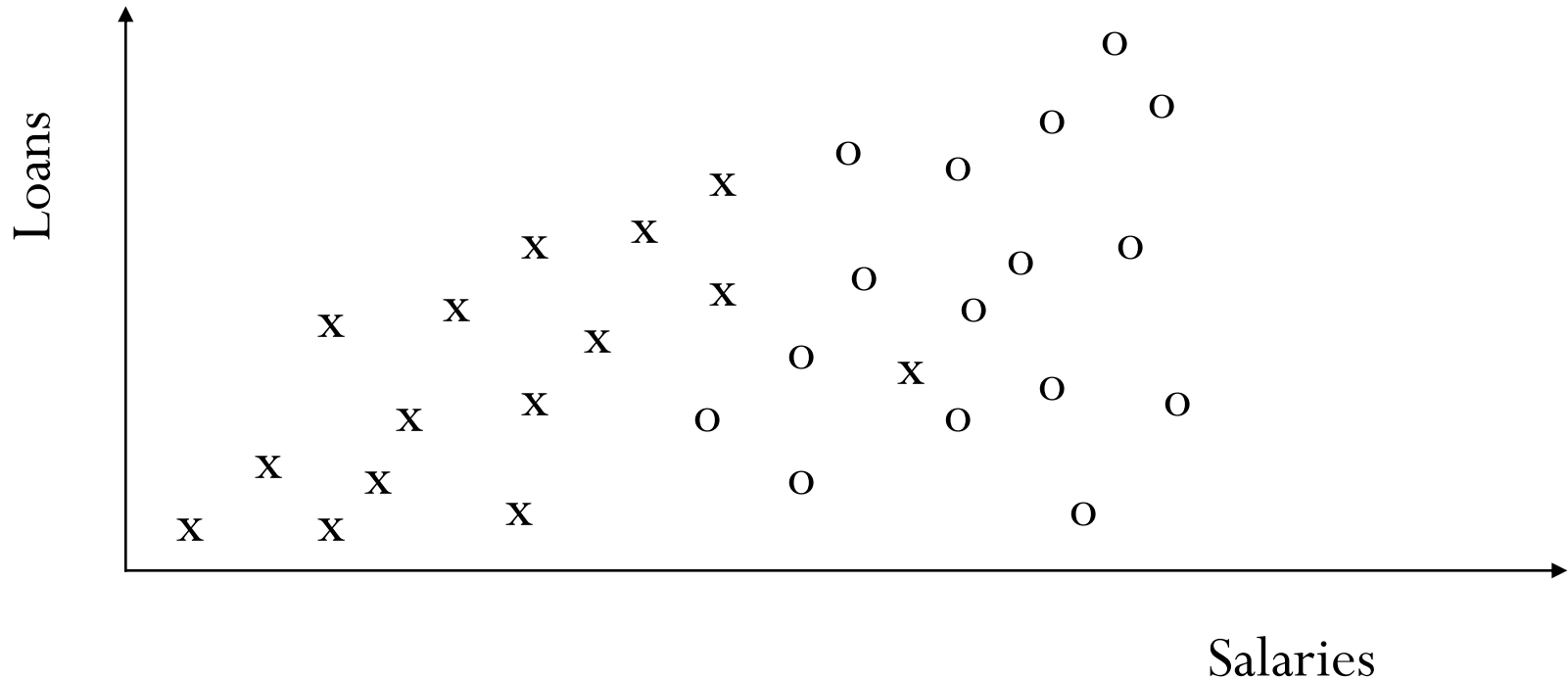


Data & Patterns

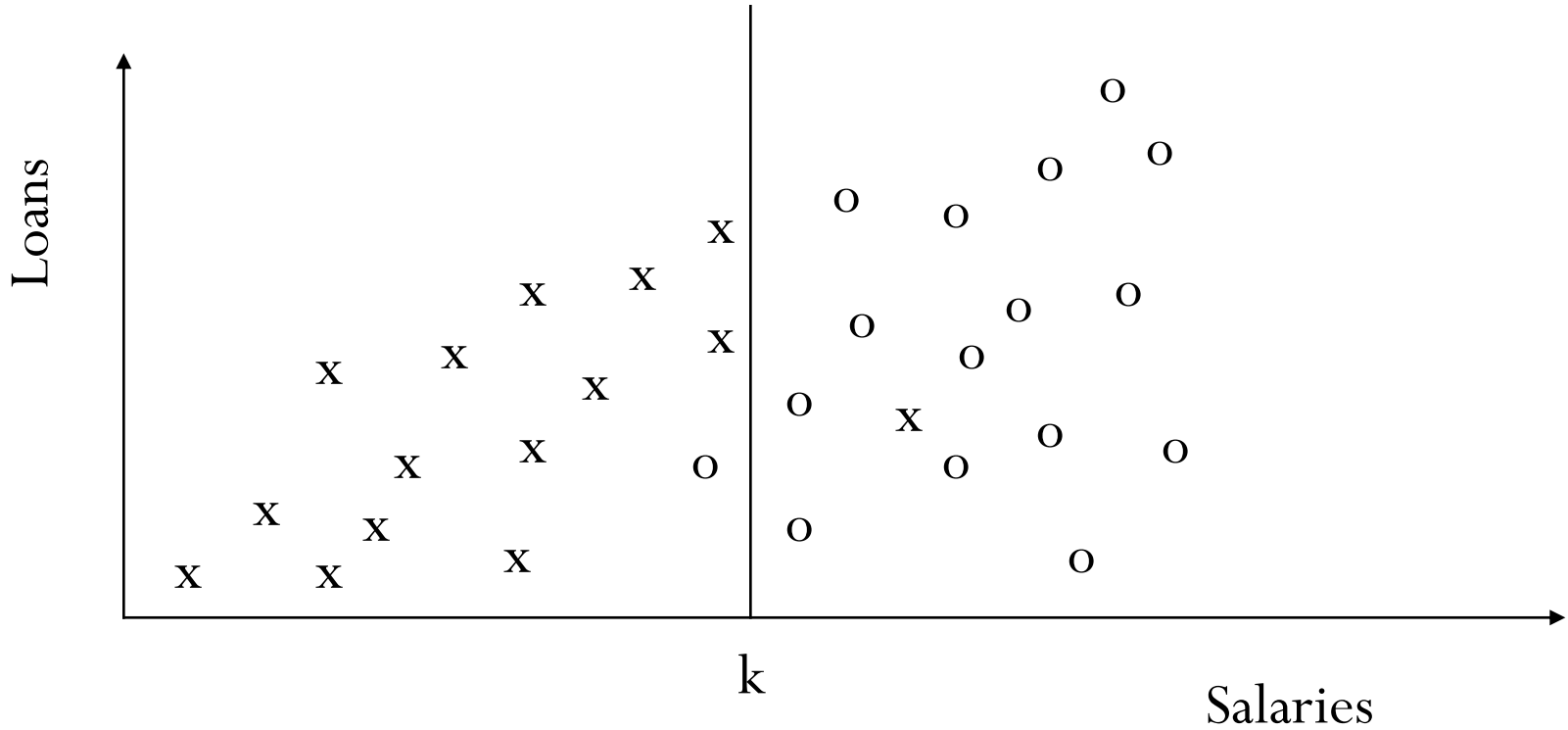
- Input: raw data
 - Big data
 - Usually unstructured
 - Coming from different sources
- Output: patterns
 - Expression, in an appropriate language, that describes succinctly some information extracted from the data
 - Features
 - Regularities on data
 - High-level information



Example



Example

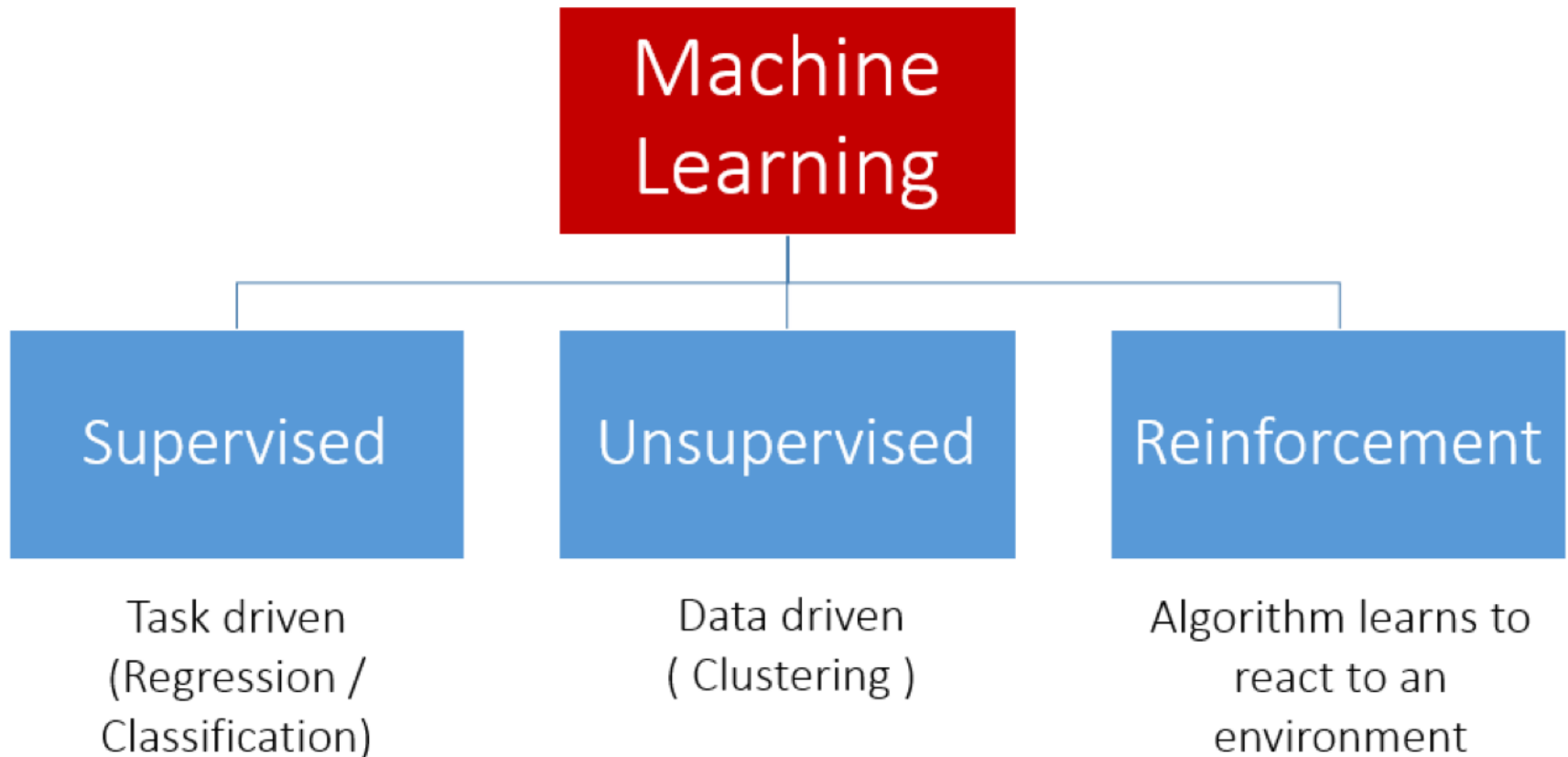


- **Pattern:** IF salary < k THEN missed payments

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



Tools

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



In Our Context...



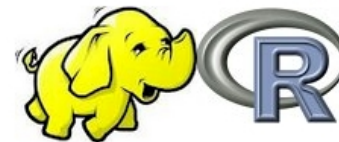
- Efficient in analyzing/mining data
- Do not scale

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

How to integrate these two worlds
together

Big data projects

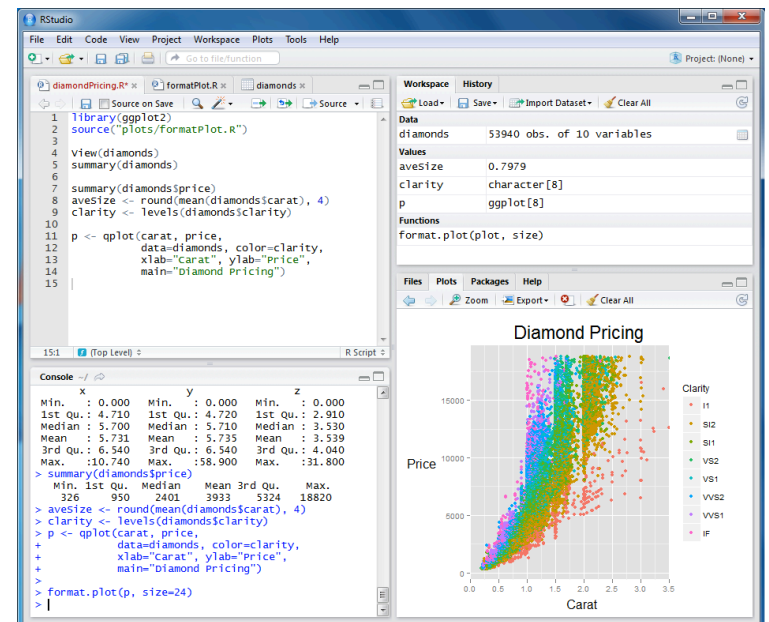
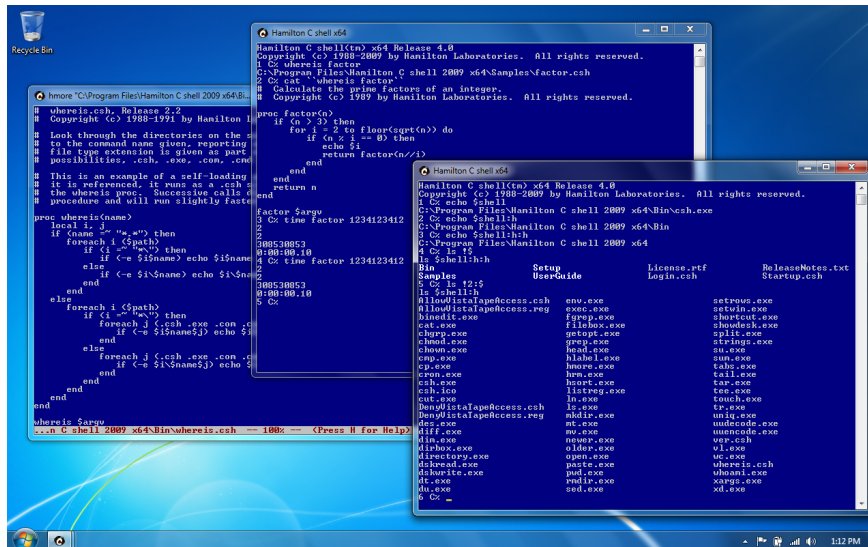
- R over a cluster computing framework
 - Rhadoop/RHive:
 - Open source extension of R on Hadoop
 - Revolution R:
 - R distribution from Microsoft
 - SparkR (R on Spark)
 - Many other connectors
- Apache Mahout
 - Open-source package on Hadoop for data mining and machine learning
- Apache MLlib
 - Spark's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

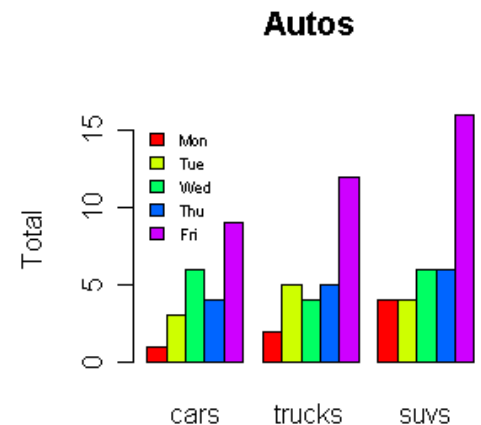
```
barplot(as.matrix(autos_data), main="Autos", ylab= "Total",  
  beside=TRUE, col=rainbow(5))
```

```
# 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)

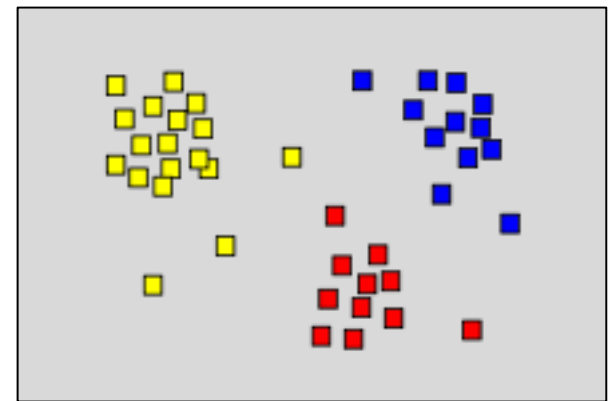
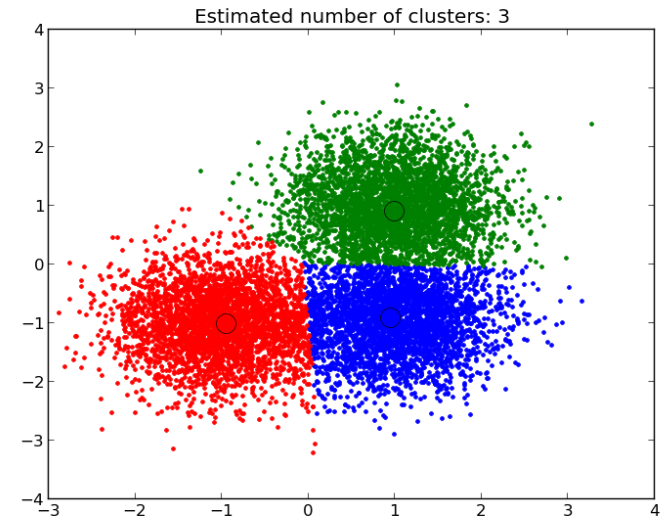
C1: Collaborative Filtering

- 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



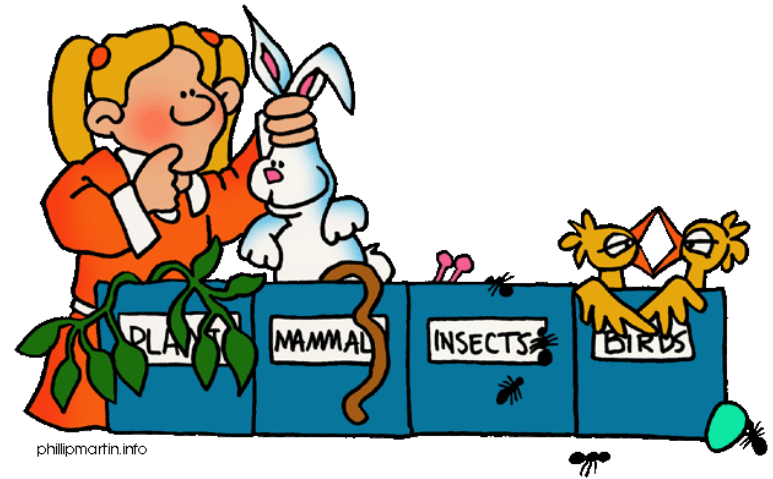
C2: Clustering

- 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,
 - ...

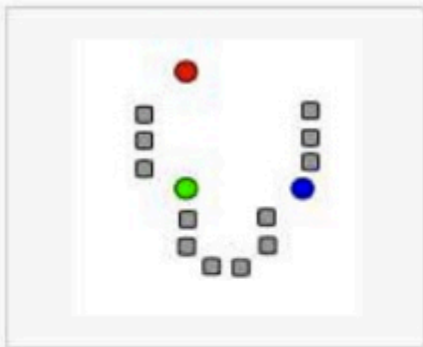


We Focus On...

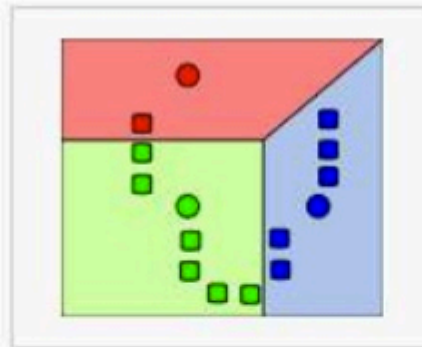
- Clustering
 - K-Means
- Classification
 - Naïve Bayes
- Frequent Pattern Mining
 - Apriori

K-Means Algorithm

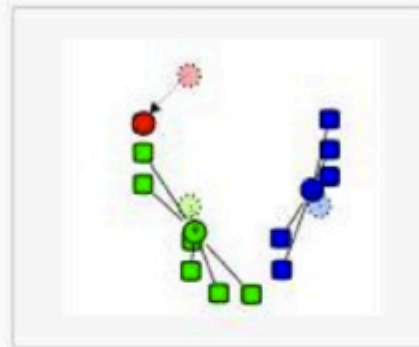
Demonstration of the standard algorithm



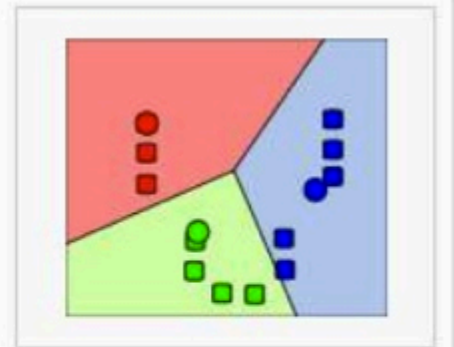
1) k initial "means" (in this case $k=3$) are randomly selected from the data set (shown in color).



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.

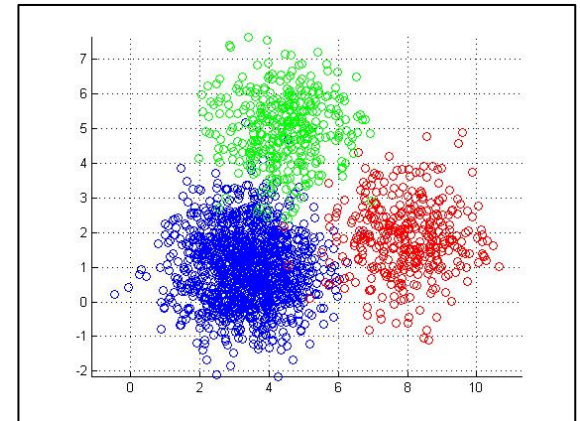
Iterative algorithm until converges

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

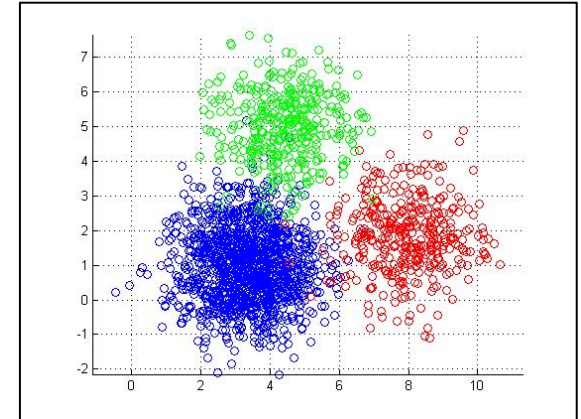
K-Means in MapReduce

- 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 $\langle \text{centroid}, \text{point} \rangle$



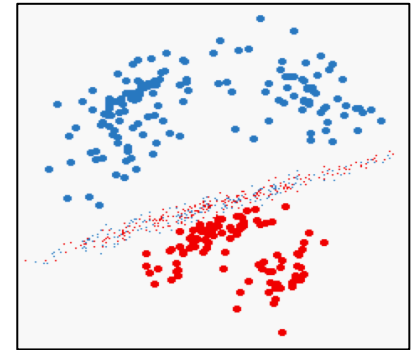
K-Means in MapReduce (Cont'd)

- 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:
 - $\langle \text{label}, \text{feature}_1, \text{feature}_2, \dots, \text{feature}_N \rangle$
 - Label: is the class label
 - Features 1..N: the features (dimensions of the point)
- Then, given a point without a label $\langle ??, \text{feature}_1, \dots, \text{feature}_N \rangle$
 - Use the model to decide on its label



Naïve Bayes Classifier: Example

Three features

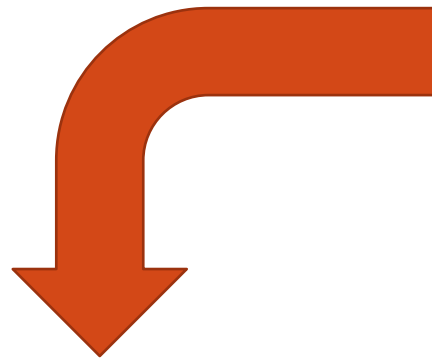
Class label (male or female)

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	6	130	8

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

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

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

Naïve Bayes: Classify New Object (Cont'd)

Male or female?



sex	height (feet)	weight (lbs)	foot size(inches)
sample 6	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
- $p(\text{feature} | \text{label}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(f - \mu)^2}{2\sigma^2}\right)$, f is the feature value in sample

Frequent Pattern Mining



- **Input:**

- a set of items $I = \{\text{milk, bread, jelly, ...}\}$
- a set of transactions where each transaction contains subset of items
 - $t1 = \{\text{milk, bread, water}\}$
 - $t2 = \{\text{milk, nuts, butter, rice}\}$

- **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:** $\# \text{trans. containing } X \cup Y / \# \text{trans. in } D$
 - Statistical relevance
- **Confidence C:** $\# \text{trans. containing } X \cup Y / \# \text{trans. containing } X$
 - Relevance of the implication

Example

Milk \Rightarrow Eggs

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

Example

TRANSACTION ID

2000

1000

4000

5000

TRANSACTIONS

A,B,C

A,C

A,D

B,E,F

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

Examples (cont)

TRANSACTION ID

2000

1000

4000

5000

TRANSACTIONS

A,B,C

A,C

A,D

B,E,F

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

Approach (cont)

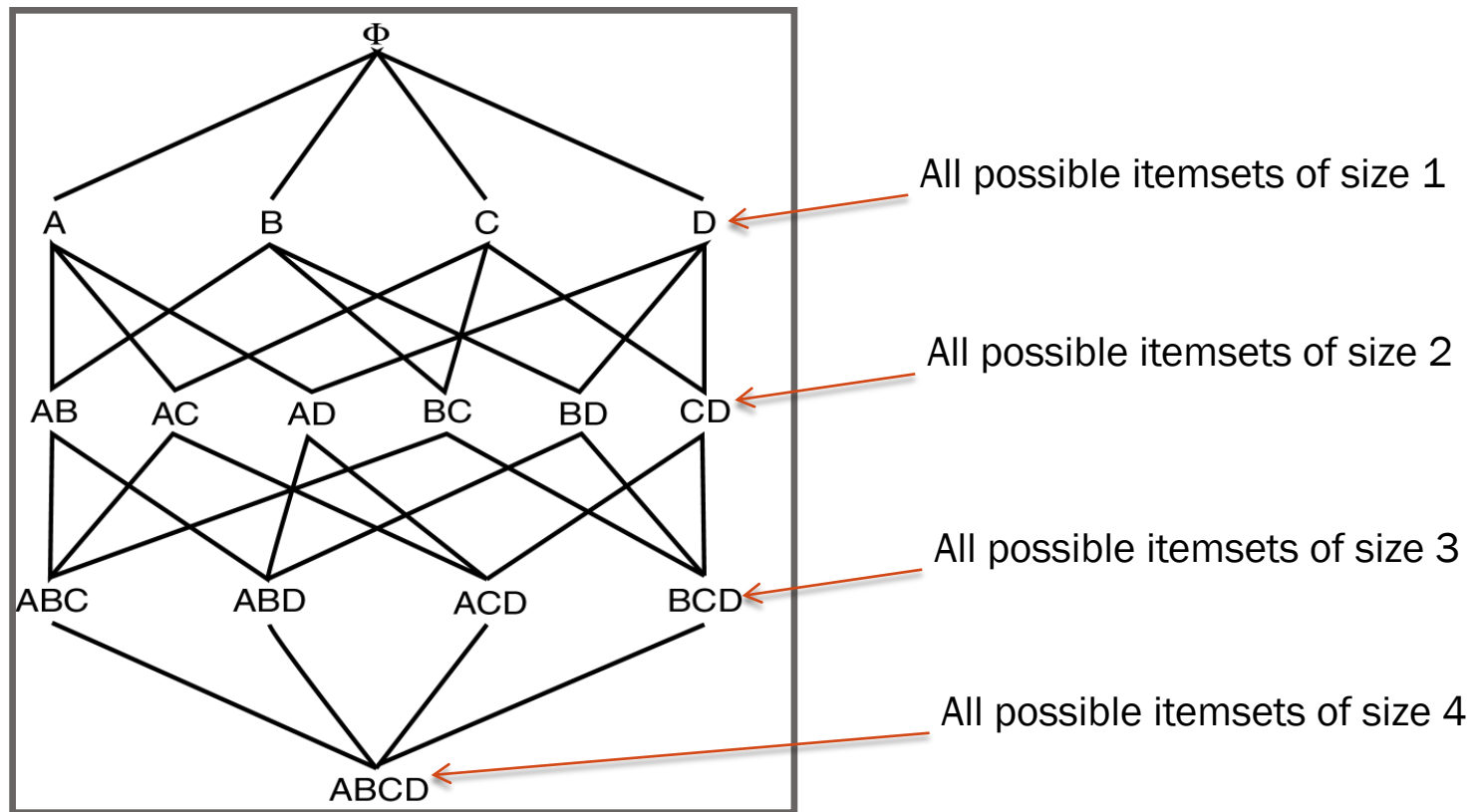
- Step 2: rule extraction
 - Minimal confidence 50%
 - Confidence of rule $A \Rightarrow C$
 - $\text{Support } \{A,C\} / \text{Support } \{A\} = 66.6\%$
 - Confidence of rule $C \Rightarrow A$
 - $\text{Support } \{A,C\} / \text{Support } \{C\} = 100\%$
 - Extracted rules
 - $A \Rightarrow C$ support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

Application

- Market basket analysis
 - $* \Rightarrow \text{eggs}$
 - What should we promote to increase the sales of eggs?
 - $\text{Milk} \Rightarrow *$
 - 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

- $\{\text{Bread}\} : 80\%$
- $\{\text{PeanutButter}\} : 60\%$
- $\{\text{Bread, PeanutButter}\} : 60\%$
- Important property:
 - For itemset $S = \{X, Y, Z, \dots\}$ 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

The screenshot displays a virtual machine environment with two main windows. The left window is a terminal showing the execution of Mahout commands and the output of a Hadoop job. The right window is a web browser displaying a blog post about Mahout.

Terminal - cloudera@localhost:

```
[cloudera@localhost ml-1m]$ cd ..
[cloudera@localhost Desktop]$ cd ..
[cloudera@localhost ~]$ mahout recommenditembased --input / --output /
--tempDir /tmp --usersFile ratings.csv
Running on hadoop, using HADOOP_HOME=/usr/lib/hadoop-0.20
HADOOP_CONF_DIR=/etc/hadoop-0.20/conf
MAHOUT-JOB: /usr/lib/mahout/mahout-examples-0.5-cdh3u3-job.jar
12/04/01 02:26:51 INFO common.AbstractJob: Command line arguments: {-b
booleanData=false, --endPhase=2147483647, --input=/, --maxCooccurrencesP
erItem=100, --maxPrefsPerUser=10, --maxSimilaritiesPerItem=100, --minPr
efsPerUser=1, --numRecommendations=10, --output=/, --similarityClassnam
e=SIMILARITY_COOCURRENCE, --startPhase=0, --tempDir=/tmp, --usersFile=
ratings.csv}
12/04/01 02:27:08 INFO input.FileInputFormat: Total input paths to proc
ess: 7
12/04/01 02:27:08 WARN snappy.LoadSnappy: Snappy native library is avai
lable
12/04/01 02:27:08 INFO util.NativeCodeLoader: Loaded the native-hadoop
library
12/04/01 02:27:08 INFO snappy.LoadSnappy: Snappy native library loaded
12/04/01 02:27:16 INFO mapred.JobClient: Running job: job_201204010150
0001
12/04/01 02:27:17 INFO mapred.JobClient: map 0% reduce 0%
```

Terminal - cloudera@localhost:/usr/share/doc/mahout-0.5+9.3/mahout-core

```
[cloudera@localhost ~]$ whereis mahout
mahout: /usr/bin/mahout /etc/mahout /usr/lib/mahout
[cloudera@localhost ~]$ sudo find / -name mahout
/usr/bin/mahout
/usr/share/doc/mahout-0.5+9.3/mahout-core/org/apache/mahout
/usr/share/doc/mahout-0.5+9.3/mahout-math/org/apache/mahout
/usr/share/doc/mahout-0.5+9.3/mahout-examples/org/apache/mahout
/usr/share/doc/mahout-0.5+9.3/mahout-utils/org/apache/mahout
/usr/lib/mahout
/etc/mahout
[cloudera@localhost ~]$ sudo find / -name mahout-core
/usr/share/doc/mahout-0.5+9.3/mahout-core
[cloudera@localhost ~]$ cd /usr/share/doc/mahout-0.5+9.3/mahout-core/
[cloudera@localhost mahout-core]$ ls
allclasses-frame.html  index.html          package-list
allclasses-noframe.html options              packages
constant-values.html   org                  resources
deprecated-list.html   overview-frame.html serialized-form.html
help-doc.html           overview-summary.html stylesheet.css
index-all.html         overview-tree.html
[cloudera@localhost mahout-core]$ sudo cp /usr/share/doc/mahout-0.5+9.3/mahout-ex
amples/org/apache/mahout/cf/taste/example/grouplens/ratings.dat .
```

Recommendation with Apache Mahout in CDH3 | Apache Hadoop for the Enterprise | Cloudera - Mozilla Firefox

http://www.cloudera.com/blog/2011/11/recommendation-with-apache-mahout-in-cdh3/

userID [itemID, score, ...]

which represents the userIDs with their recommended itemIDs along with the preference scores. This algorithm is also implemented in MapReduce and the following table has some of the options we can specify for the job:

Flag	Description
--input (-i)	Path to the job input directory in HDFS (-i shortcut flag is broken in 0.5, MAHOUT-842)
--output (-o)	Path to the output dir in HDFS
--usersFile	File listing users to generate recommendations for
--tempDir	Intermediary output directory

To set things up we need to setup the list of users we want to produce a recommendation for. Put the number "6040" (a user ID in our source data) on a line by itself in a file and put that in HDFS. Use the hadoop command line tools to copy this file to HDFS:

```
hadoop fs -put [my_local_file] [user_file_location_in_hdfs]
```

With our user list in hdfs we can now run the Mahout recommendation job with a command in the form of:

```
mahout recommenditembased --input [input-hdfs-path] --output [output-hdfs-path] --te
```

which will run for a while (a chain of 10 MapReduce jobs) and then write out the item recommendations into HDFS we can now take a look at. If we tail the output from the RecommenderJob with the command:

```
hadoop fs -cat [output-hdfs-path]/part-r-00000
```

it should look like:

```
6040 [1941:5.0, 1904:5.0, 2859:5.0, 3811:5.0, 3814:5.0, 14:5.0, 17:5.0, 3795:5.0, 3794:5
```

where each line represents a userID with associated recommended itemIDs and their scores.

Conclusion

This concludes the introduction to Mahout's implementation of Collaborative Filtering which is now included in CDH3u2. CF is but one of the many techniques included in the Mahout project and the reader should now be ready to not only further explore CF but tackle some of the other techniques as well. Special thanks to Sean Owen for review help on this article.

References

- S. Owen, R. Anil, T. Dunning, E. Friedman: Mahout in Action
- Sarwar et al.: Item-Based Collaborative Filtering Recommendation Algorithms
- Apache Mahout Wiki:

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

The screenshot shows an IDE with a project named 'spam_classifier'. The main editor displays the 'spam_classifier.scala' file, which contains the following code:

```
39  val ham = sc.textFile("ham.txt")
40
41  // (1) Feature extraction
42  // Create a HashingTF instance to map email text to vectors of 100 features.
43  val tf = new HashingTF(numFeatures = 100)
44  // Each email is split into words, and each word is mapped to one feature.
45  val spamFeatures = spam.map(email => tf.transform(email.split(" ")))
46  val hamFeatures = ham.map(email => tf.transform(email.split(" ")))
47
48  // (2) Create Training Set
49  // Create LabeledPoint datasets for positive (spam) and negative (ham) examples.
50  val positiveExamples = spamFeatures.map(features => LabeledPoint(1, features))
51  val negativeExamples = hamFeatures.map(features => LabeledPoint(0, features))
52  val trainingData = positiveExamples ++ negativeExamples
53  trainingData.cache() // Cache data since Logistic Regression is an iterative algorithm.
54
55  // (3) Training
56  // Create a Logistic Regression learner which uses the LBFGS optimizer.
57  val lrLearner = new LogisticRegressionWithSGD()
```

The bottom panel shows the output of the program, which includes the following log messages:

```
15/07/13 11:39:47 INFO TaskSetManager: Finished task 2.0 in stage 102.0 (TID 407) in 5 ms on localhost (3/4)
15/07/13 11:39:47 INFO TaskSetManager: Finished task 3.0 in stage 102.0 (TID 408) in 6 ms on localhost (4/4)
15/07/13 11:39:47 INFO TaskSchedulerImpl: Removed TaskSet 102.0, whose tasks have all completed, from pool
15/07/13 11:39:47 INFO DAGScheduler: ResultStage 102 (treeAggregate at GradientDescent.scala:189) finished in 0.009 s
15/07/13 11:39:47 INFO DAGScheduler: Job 102 finished: treeAggregate at GradientDescent.scala:189, took 0.015900 s
15/07/13 11:39:47 INFO GradientDescent: GradientDescent.runMiniBatchSGD finished. Last 10 stochastic losses 0.08569670501181827, 0.08555174646287429, 0.08540982506499899, 0.0852708
Prediction for positive test example: 1.0
Prediction for negative test example: 0.0
15/07/13 11:39:47 INFO SparkUI: Stopped Spark web UI at
15/07/13 11:39:47 INFO DAGScheduler: Stopping DAGScheduler
15/07/13 11:39:47 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
15/07/13 11:39:47 INFO Utils: path = C:\Users\Robin\AppData\Local\Temp\spark-e8892951-15d0-40db-afd9-5915b906ca98\blockmgr-88f4aebd-8033-41c3-8902-fcc60096e5ca, already present as
15/07/13 11:39:47 INFO MemoryStore: MemoryStore cleared
15/07/13 11:39:47 INFO BlockManager: BlockManager stopped
15/07/13 11:39:47 INFO BlockManagerMaster: BlockManagerMaster stopped
15/07/13 11:39:47 INFO SparkContext: Successfully stopped SparkContext
15/07/13 11:39:47 INFO Utils: Shutdown hook called
15/07/13 11:39:47 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
15/07/13 11:39:47 INFO Utils: Deleting directory C:\Users\Robin\AppData\Local\Temp\spark-e8892951-15d0-40db-afd9-5915b906ca98
15/07/13 11:39:47 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.
```

The output also shows the predictions for the test examples: "Prediction for positive test example: 1.0" and "Prediction for negative test example: 0.0".