Fuzzy Models for Data Science and Big Data

Francisco Herrera and Alberto Fernández
Soft Computing and Information Intelligent Systems (SCI²S)
University of Granada, Spain
Email: {herrera, alberto}@decsai.ugr.es
http://sci2s.ugr.es

Isaac Triguero
School of Computer Science
Automated Scheduling Optimisation and Planning (ASAP)
University of Nottingham
United Kingdom
Isaac.Triguero@nottingham.ac.uk
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Big Data

Our world revolves around the data

- **Science**
  - Data bases from astronomy, genomics, environmental data, transportation data, ...

- **Humanities and Social Sciences**
  - Scanned books, historical documents, social interactions data, ...

- **Business & Commerce**
  - Corporate sales, stock market transactions, census, airline traffic, ...

- **Entertainment**
  - Internet images, Hollywood movies, MP3 files, ...

- **Medicine**
  - MRI & CT scans, patient records, ...

- **Industry, Energy, ...**
  - Sensors, ...
What is Big Data?

There is not a standard definition!

“Big Data” involves data whose volume, diversity and complexity requires new techniques, algorithms and analyses to extract valuable knowledge (hidden).
What is Big Data? The 5V’s definition

- Volume
- Velocity
- Variety
- Veracity

Value
Big data refers to any problem characteristic that represents a challenge to process it with traditional applications.
Big data has many faces
How to deal with data intensive applications? **Scale-up vs. Scale-out**
Traditional HPC way of doing things

Communications network (Infiniband)

- Worker nodes (lots of them)
- Lots of computations
- Lots of communication

Network for I/O

- Limited I/O

Central storage

- Input data (relatively small)

Source: Jan Fostier. Introduction to MapReduce and its Application to Post-Sequencing Analysis
Data-intensive jobs

- Fast communication network (Infiniband)
- Limited communication
- Low compute intensity
- Doesn’t scale
- Lots of I/O

Network for I/O

Central storage

Input data (lots of it)
Data-intensive jobs

Communication network

Limited communication

Low compute intensity

input data (lots of it)

Solution: store data on local disks of the nodes that perform computations on that data ("data locality")
Distributed systems in Big Data

**Objective:** To apply an operation to all data

- One machine cannot process or store all data
  - Data is distributed in a cluster of computing nodes
  - It does not matter which machine executes the operation
  - It does not matter if it is run twice in different nodes (due to failures or straggler nodes)
  - We look for an abstraction of the complexity behind distributed systems

**DATA LOCALITY is crucial**
- Avoid data transfers between machines as much as possible
New programming model: **MapReduce**

- “Moving computation is cheaper than moving computation and data at the same time”

**Idea**

- Data is distributed among nodes (distributed file system)
- Functions/operations to process data are distributed to all the computing nodes
- Each computing node works with the data stored in it
- Only the necessary data is moved across the network
MapReduce

- Parallel Programming model
- **Divide & conquer strategy**
  - *divide*: partition dataset into smaller, independent chunks to be processed in parallel (*map*)
  - *conquer*: combine, merge or otherwise aggregate the results from the previous step (*reduce*)
- Based on **simplicity** and **transparency** to the programmers, and assumes **data locality**
- Becomes popular thanks to the open-source project Hadoop! (Used by Google, Facebook, Amazon, ...)

15
MapReduce: How it works
MapReduce: WordCount Example

**Input File**
- Hello World Bye World
- Hello Hadoop Goodbye Hadoop

**Map key Value Splitting**
- Hello, 1
  - World, 1
  - Bye, 1
  - World, 1
- Hello, 1
  - Hadoop, 1
  - Goodbye, 1
  - Hadoop, 1

**Short and Shuffle**
- Hello, 1
- World, 1
- Bye, 1
- Hadoop, 1
- Goodbye, 1

**Reduce key Value Pairs**
- Hello, {1,1}
- World, {1,1}
- Bye, {1}
- Hadoop, {1,1}
- Goodbye, {1,1}

**Output**
- Hello, 2
  - World, 2
  - Bye, 1
  - Hadoop, 2
  - Goodbye, 1
Hadoop

- **Open source framework for Big Data processing**
  - Based on two Works published by Google
    - **Google File System** (GFS)[Ghe03]
    - **MapReduce** algorithm[Dea04]
  - Composed of
    - **Hadoop Distributed File System** (HDFS) ➔ Storage
    - Implementation of the MapReduce algorithm ➔ Processing

---

Hadoop

- Hadoop is:
  - An **open-source** framework written in Java
  - Distributed storage of very large data sets (Big Data)
  - Distributed processing of very large data sets

- This framework consists of a **number of modules**
  - *Hadoop Common*
  - *Hadoop Distributed File System (HDFS)*
  - *Hadoop YARN* – resource manager
  - *Hadoop MapReduce* – programming model

http://hadoop.apache.org/
Hadoop: A master/slave architecture

- **Master**: NameNode, JobTracker
- **Slave**: {DataNode, TaskTracker}, ..., {DataNode, TaskTracker}
Distributed File System: HDFS

- **HDFS – Hadoop Distributed File System**
  - Distributed File System written in Java
  - Scales to clusters with **thousands of computing nodes**
    - Each node stores part of the data in the system
  - **Fault tolerant** due to data replication
  - Designed for big files and low-cost hardware
    - GBs, TBs, PBs
  - **Efficient for read and append operations** (random updates are rare)
  - High throughput (for bulk data) more important than low latency
Hadoop MapReduce: Main Characteristics

- **Automatic parallelization:**
  - Depending on the size of the input data ➔ there will be multiple MAP tasks!
  - Depending on the number of Keys <k, value> ➔ there will be multiple REDUCE tasks!

- **Scalability:**
  - It may work over every data center or cluster of computers.

- **Transparent for the programmer**
  - Fault-tolerant mechanism.
  - Automatic communications among computers
Data Sharing in Hadoop MapReduce

- Input
- HDFS read
- iter. 1
- HDFS write
- iter. 2
- HDFS read
- HDFS write
- ...

Input

HDFS read

query 1 → result 1

query 2 → result 2

query 3 → result 3

Slow due to replication, serialization, and disk IO
Paradigms that do not fit with Hadoop MapReduce

- Directed Acyclic Graph (DAG) model:
  - The DAG defines the dataflow of the application, and the vertices of the graph defines the operations on the data

- Graph model:
  - More complex graph models that better represent the dataflow of the application
  - Cyclic models -> Iterativity.

- Iterative MapReduce model:
  - An extended programming model that supports iterative MapReduce computations efficiently
New platforms to overcome Hadoop’s limitations

- **GIRAPH (APACHE Project)**
  ![GIRAPH](http://giraph.apache.org/)
  *Iterative graph processing*

- **GPS - A Graph Processing System, (Stanford)**
  ![GPS](http://infolab.stanford.edu/gps/)
  *Amazon's EC2*

- **Distributed GraphLab (Carnegie Mellon Univ.)**
  ![GraphLab](https://github.com/graphlab-code/graphlab)
  *Amazon’s EC2*

- **Twister (Indiana University)**
  ![Twister](http://www.iterativemapreduce.org/)
  *Private Clusters*

- **PrIter (University of Massachusetts Amherst, Northeastern University-China)**
  ![PrIter](http://code.google.com/p/priter/)
  *Private cluster and Amazon EC2 cloud*

- **HaLoop (University of Washington)**
  ![HaLoop](http://clue.cs.washington.edu/node/14)
  ![HaLoop](http://code.google.com/p/haloop/)
  *Amazon’s EC2*

- **Spark (UC Berkeley)**
  ![Spark](http://spark.incubator.apache.org/research.html)
  *100 times more efficient than Hadoop, including iterative algorithms, according to creators*

- **GPU based platforms**
  - **Mars**
  - **Grex**
  ![Mars](http://www.nvidia.com/)
  ![Grex](http://www.nvidia.com/)

---

25
Big data technologies

The World of Big Data Tools

- DAG Model
  - Hadoop
- MapReduce Model
- Graph Model
  - MPI
    - HaLoop
    - Twister
    - Spark
    - Giraph
    - Hama
    - GraphLab
    - GraphX
  - Harp
    - Stratosphere
    - Reef
  - Dryad/DryadLINQ
    - Pig/PigLatin
    - Hive
    - Tez
    - Shark
    - MRQL
- BSP/Collective Model

For Iterations/Learning

- Spark
- Harp
- Stratosphere
- Reef

For Query

- Drill
  - Pig/PigLatin
  - Hive
  - Tez
  - Shark

For Streaming

- S4
- Storm
- Samza
  - Spark Streaming
What is Spark?

Fast and Expressive Cluster Computing Engine Compatible with Apache Hadoop

Efficient
- General execution graphs
- In-memory storage

Usable
- Rich APIs in Java, Scala, Python
- Interactive shell

Up to 10\times faster on disk, 100\times in memory

2-5\times less code
What is Spark?

- Data processing engine (only)
- Without a distributed file system
  - Uses other existing DFS
    • HDFS, NoSQL...
    • Hadoop is not a prerequisite
- Works with different cluster management tools
  - Hadoop (YARN)
  - Mesos
  - Standalone mode (included in Spark)
What is Spark?

Spark SQL structured data
Spark Streaming real-time
MLib machine learning
GraphX graph processing

Spark Core

Standalone Scheduler
YARN
Mesos
Spark Goal

- Provide distributed memory abstractions for clusters to support apps with working sets
- **Retain the attractive properties of MapReduce:**
  - Fault tolerance (for crashes & stragglers)
  - Data locality
  - Scalability

**Solution:** augment data flow model with “resilient distributed datasets” (RDDs)
RDDs in Detail

- An RDD is a fault-tolerant collection of elements that can be operated on in parallel.

- There are two ways to create RDDs:
  - Parallelizing an existing collection in your driver program
  - Referencing a dataset in an external storage system, such as a shared filesystem, HDFS, Hbase.

- *Can be cached for future reuse*
Operations with RDDs

- Transformations (e.g. map, filter, groupBy, join)
  - Lazy operations to build RDDs from other RDDs
- Actions (e.g. count, collect, save)
  - Return a result or write it to storage

<table>
<thead>
<tr>
<th>Transformations (define a new RDD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
</tr>
<tr>
<td>filter</td>
</tr>
<tr>
<td>sample</td>
</tr>
<tr>
<td>union</td>
</tr>
<tr>
<td>groupByKey</td>
</tr>
<tr>
<td>reduceByKey</td>
</tr>
<tr>
<td>join</td>
</tr>
<tr>
<td>cache</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parallel operations (return a result to driver)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduce</td>
</tr>
<tr>
<td>collect</td>
</tr>
<tr>
<td>count</td>
</tr>
<tr>
<td>save</td>
</tr>
<tr>
<td>lookupKey</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Operations with RDDs

- **RDDs – Simple example**

```python
>>> lines = sc.textFile("README.md")  # Creates an RDD
>>> lines.count()  # Counts the number of elements in the RDD
127
>>> lines.first()  # 1st element of the RDD -> 1st line of README.md
u'# Apache Spark'
```

- **Simple wordCount in Spark**

```python
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" "))
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```
Spark vs. hadoop

Lines of code for K-Means

Spark ~ 90 lines –

Hadoop ~ 4 files, > 300 lines

Spark

- **Driver** and **Workers**
  - A Spark program is composed of two programs
    - **Driver program**
    - **Workers program**
      - Executed in the computing nodes
      - Or in local threads
  - RDDs are distributed across the whole cluster
SparkSQL: Datasets & DataFrames

Structured APIs

- **DataFrames (>= Spark 1.3)**
  - Idea: **RDDs of rows with columns that can be accessed by their names**
  - **Similar to Pandas in Python (dataframes in R)**
  - Avoid Java serialization performed by RDDs
  - API natural for developers familiar with building query plans (SQL)
  - Introduced as a part of Tungsten project
    - Efficient memory management
  - Concept of schema to describe data
SparkSQL: Datasets & DataFrames

- **Structured APIs**
  - **Datasets (>= Spark 1.6)**
    - Idea: Strongly typed RDDs
    - **Functional transformations** (map, flatMap, filter)
    - **Best of both RDDs and DataFrames**
      - Object-oriented programming
      - Compile-time type safety
      - Catalyst optimization
      - Tungsten off-heap memory optimization
  - Only for Scala and Java
SparkSQL: Datasets y DataFrames

- **Structured APIs**
  - **DataFrames and Datasets**
  - **Fused in Spark 2.0 (November 2016)**
    - A DataFrame is just a Dataset of Rows: Dataset[Row]
  - Both make use of Catalyst and Tungsten projects
## SparkSQL: Datasets vs DataFrames

### Structured APIs in Spark
- Analysis of the *reported errors* before a job is executed

<table>
<thead>
<tr>
<th></th>
<th>SQL</th>
<th>DataFrames</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax Errors</td>
<td>Runtime</td>
<td>Compile Time</td>
<td>Compile Time</td>
</tr>
<tr>
<td>Analysis Errors</td>
<td>Runtime</td>
<td>Runtime</td>
<td>Compile Time</td>
</tr>
</tbody>
</table>
Apache Flink is an open source platform for distributed stream and batch data processing.

Flink’s core is a streaming dataflow engine that provides data distribution, communication, and fault tolerance for distributed computations over data streams. Flink includes several APIs for creating applications that use the Flink engine:

1. DataStream API for unbounded streams embedded in Java and Scala, and
2. DataSet API for static data embedded in Java, Scala, and Python,
3. Table API with a SQL-like expression language embedded in Java and Scala.

Flink also bundles libraries for domain-specific use cases:

1. CEP, a complex event processing library,
2. Machine Learning library, and
3. Gelly, a graph processing API and library.

You can integrate Flink easily with other well-known open source systems both for data input and output as well as deployment.

Streaming First

High throughput and low latency stream processing with exactly-once guarantees.

Throughput

<table>
<thead>
<tr>
<th># CPU Cores</th>
<th>Flink (millions)</th>
<th>Storm (millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>150</td>
<td>3</td>
</tr>
</tbody>
</table>

Batch on Streaming

Batch processing applications run efficiently as special cases of stream processing applications.

APIs, Libraries, and Ecosystem

DataStream, DataSet, and more. Integrated with the Apache Big Data stack.
Big Data: Technology and Chronology

2001
3V’s Gartner
Doug Laney

2004
MapReduce
Google
Jeffrey Dean

2009-2013 Flink
TU Berlin
Flink Apache (Dec. 2014) Volker Markl

2010 Spark
U Berckeleay
Apache Spark
Feb. 2014
Matei Zaharia

2008 Hadoop
Yahoo!
Doug Cutting

2010-2017:
Big Data Analytics: Mahout, MLLib, ...
Hadoop Ecosystem
Applications
New Technology

2010-2017:
Big Data Ecosystem

- **Apache Mesos**: Cluster management
- **kubernetes**: Cluster management
- **ZooKeeper**: Coordinador
- **AVRO**: Data serialization
- **Hadoop**: to RDBMS
- **Pig**: Scripting para MapReduce
- **SQOOP**: Hadoop to RDBMS
- **HIVE**: SQL
- ** cassandra**: NoSQL Database
- **STORM**: Streaming data
- **Flink**: Streaming data
- **DP**: DAG execution
- **Apache HBase**: NoSQL on HDFS
- **Kafka**: Gestión de fuentes de datos
- **Impala**: Interactive SQL
- **Zepelin**: Web-based notebook
- **Docker**: Containers
- **Flume**: Log data
Take-home message so far

- We need new strategies to deal with big datasets
  - Choosing the right technology is like choosing the right data structure in a program.
- The world of big data is rapidly changing. Being up-to-date is difficult but necessary.
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Big Data Analytics

Potential scenarios:

Clustering

Classification

Real Time Analytics/Big Data Streams

Association

Recommendation Systems

Social Media Mining

Social Big Data
Machine learning for Big Data

- Data mining techniques have demonstrated to be very useful tools to extract new valuable knowledge from data.
- The knowledge extraction process from big data has become a very difficult task for most of the **classical** and **advanced** data mining tools.
- The main challenges are to deal with:
  - The increasing scale of data
    - at the level of **instances**
    - at the level of **features**
  - The **complexity** of the problem.
  - And many other points
# Big Data Analytics: A 3 generational view

<table>
<thead>
<tr>
<th>Generation</th>
<th>1st Generation</th>
<th>2nd Generation</th>
<th>3rd Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>SAS, R, Weka, SPSS, KEEL</td>
<td>Mahout, Pentaho, Cascading</td>
<td>Spark, Haloop, GraphLab, Pregel, Giraph, ML over Storm</td>
</tr>
<tr>
<td>Scalability</td>
<td>Vertical</td>
<td>Horizontal (over Hadoop)</td>
<td>Horizontal (Beyond Hadoop)</td>
</tr>
<tr>
<td>Algorithms</td>
<td>Huge collection of algorithms</td>
<td>Small subset: sequential logistic regression, linear SVMs, Stochastic Gradient Descendent, k-means clustering, Random forest, etc.</td>
<td>Much wider: CGD, ALS, collaborative filtering, kernel SVM, matrix factorization, Gibbs sampling, etc.</td>
</tr>
<tr>
<td>Algorithms Not Available</td>
<td>Practically nothing</td>
<td>Vast no.: Kernel SVMs, Multivariate Logistic Regression, Conjugate Gradient Descendent, ALS, etc.</td>
<td>Multivariate logistic regression in general form, k-means clustering, etc. – Work in progress to expand the set of available algorithms</td>
</tr>
<tr>
<td>Fault-Tolerance</td>
<td>Single point of failure</td>
<td>Most tools are FT, as they are built on top of Hadoop</td>
<td>FT: HaLoop, Spark Not FT: Pregel, GraphLab, Giraph</td>
</tr>
</tbody>
</table>
Mahout (Samsara)

- First ML library initially based on Hadoop MapReduce.
- Abandoned MapReduce implementations from version 0.9.
- Nowadays it is focused on a new math environment called Samsara.
- It is integrated with Spark, Flink and H2O.
- Main algorithms:
  - Stochastic Singular Value Decomposition (ssvd, dssvd)
  - Stochastic Principal Component Analysis (spca, dspca)
  - Distributed Cholesky QR (thinQR)
  - Distributed regularized Alternating Least Squares (dals)
  - Collaborative Filtering: Item and Row Similarity
  - Naive Bayes Classification

http://mahout.apache.org/
Spark Libraries

https://spark.apache.org/docs/latest/mllib-guide.html

MLlib

Spark Streaming
HDFS
Databases
Dashboards

Spark
SQL

Meta Store
HiveQL
UDFs
SerDes

Spark SQL
Apache Spark
Machine Learning Library (MLlib) Guide

MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives, as outlined below:

- Data types
- Basic statistics
  - summary statistics
  - correlations
  - stratified sampling
  - hypothesis testing
  - random data generation
- Classification and regression
  - linear models (SVMs, logistic regression, linear regression)
  - naive Bayes
  - decision trees
  - ensembles of trees (Random Forests and Gradient-Boosted Trees)
  - isotonic regression
- Collaborative filtering
  - alternating least squares (ALS)
- Clustering
  - k-means
  - Gaussian mixture
  - power iteration clustering (PIC)
  - latent Dirichlet allocation (LDA)
  - streaming k-means
- Dimensionality reduction
  - singular value decomposition (SVD)
  - principal component analysis (PCA)
- Feature extraction and transformation
- Frequent pattern mining
  - FP-growth
- Optimization (developer)
  - stochastic gradient descent
  - limited-memory BFGS (L-BFGS)

MLlib is under active development. The APIs marked experimental/developer API may change in future releases, and the migration guide below
FlinkML - Machine Learning for Flink

FlinkML is the Machine Learning (ML) library for Flink. It is a new effort in the Flink community, with a growing list of algorithms and contributors. With FlinkML we aim to provide scalable ML algorithms, an intuitive API, and tools that help minimize glue code in end-to-end ML systems. You can see more details about our goals and where the library is headed in our vision and roadmap here.

**Supported Algorithms**

- Supervised Learning
- Unsupervised Learning
- Data Preprocessing
- Recommendation
- Outlier selection
- Utilities

**Getting Started**

- Pipelines

**How to contribute**

FlinkML

Supported Algorithms
FlinkML currently supports the following algorithms:

Supervised Learning
- SVM
- Multiple linear regression
- Optimization Framework

Unsupervised Learning
- k-Nearest neighbors join

Data Preprocessing
- Polynomial Features
- Standard Scaler
- MinMax Scaler

Recommendation
- Alternating Least Squares (ALS)

Utilities
- Distance Metrics
- Cross Validation
H₂O library

H₂O Prediction Engine

<table>
<thead>
<tr>
<th>SDK/API</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapids Query R-engine</td>
<td>Nano Fast Scoring Engine</td>
</tr>
</tbody>
</table>

- In-Mem Map Reduce
  Distributed fork/join
- Memory Manager
  Columnar Compression

Deep Learning

- Cluster
- Classify
- Regression
- Trees
- Boosting
- Forests
- Solvers
- Gradients

Ensembles

Support for R, Python, Hadoop and Spark

- It contains Deep Learning algorithms
  - World record to solve the MNIST problem without preprocessing

http://www.h2o.ai/

http://0xdata.com/blog/2015/02/deep-learning-performance/
Two main ways for learning a model in Big Data:

- **Locally**
  - A model is created for each partition of the data (only using the data of that partition)
  - All the models are combined when predicting the class of a new example → Ensemble

- **Globally**
  - A single model is created using all the available data
  - They try to obtain the same model as the one that would be obtained if the method could be executed in a single node
Local model

- **Advantages**
  - Usually faster
  - Gets faster as the number of partitions is increased
  - Any existing model can be applied
  - Only the aggregation phase has to be designed

- **Disadvantages**
  - **Slow in test phase**, too many models have to be executed
  - **Loss of accuracy** as the number of partitions increases
    - With few partitions, accuracy can improve due to the ensemble effect
    - With too many partitions, the accuracy tends to drop, since there are not enough examples in each partition
  - **They do not take advantage of the data as a whole**
Machine Learning in Big Data: Global vs. Local

Global model

- **Advantages**
  - Greater **accuracy** is expected (not proved)
  - All the examples are used to learn a single model
  - Anyway, a global ensemble can also be built
  - The model is independent of the number of partitions
  - Faster in test phase

- **Disadvantages**
  - More complex design and implementation
  - Distributed nature of Big Data processing has to be taken into account (computation/communication)
Case of Study: Random Forests

- Random Forest is a very well-known machine learning technique for **classification** or regression.
  - Ensemble learning
  - Tree-based models
  - Random selection of features

- Most promising characteristics:
  - Great generalization capabilities
  - Detect variable importance
  - Relatively efficient on large databases.
Random Forest under MapReduce

1) Building phase
Random Forest under MapReduce

2) Testing phase
Random Forest under MapReduce

Case of Study: Random Forest for KddCup’99

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>972.781</td>
</tr>
<tr>
<td>DOS</td>
<td>3.883.370</td>
</tr>
<tr>
<td>PRB</td>
<td>41.102</td>
</tr>
<tr>
<td>R2L</td>
<td>1.126</td>
</tr>
<tr>
<td>U2R</td>
<td>52</td>
</tr>
</tbody>
</table>

Time elapsed (seconds) for sequential versions:

<table>
<thead>
<tr>
<th>Datasets</th>
<th>10%</th>
<th>50%</th>
<th>full</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOS versus normal</td>
<td>6344.42</td>
<td>49134.78</td>
<td>NC</td>
</tr>
<tr>
<td>DOS versus PRB</td>
<td>4825.48</td>
<td>28819.03</td>
<td>NC</td>
</tr>
<tr>
<td>DOS versus R2L</td>
<td>4454.58</td>
<td>28073.79</td>
<td>NC</td>
</tr>
<tr>
<td>DOS versus U2R</td>
<td>3848.97</td>
<td>24774.03</td>
<td>NC</td>
</tr>
<tr>
<td>normal versus PRB</td>
<td>468.75</td>
<td>6011.70</td>
<td>NC</td>
</tr>
<tr>
<td>normal versus R2L</td>
<td>364.66</td>
<td>4773.09</td>
<td>14703.55</td>
</tr>
<tr>
<td>normal versus U2R</td>
<td>295.64</td>
<td>4785.66</td>
<td>14635.36</td>
</tr>
</tbody>
</table>
Random Forest under MapReduce

**Case of Study: Random Forest for KddCup’99**

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>972.781</td>
</tr>
<tr>
<td>DOS</td>
<td>3.883.370</td>
</tr>
<tr>
<td>PRB</td>
<td>41.102</td>
</tr>
<tr>
<td>R2L</td>
<td>1.126</td>
</tr>
<tr>
<td>U2R</td>
<td>52</td>
</tr>
</tbody>
</table>

Time elapsed (seconds) for Big data versions with 20 partitions:

<table>
<thead>
<tr>
<th>Datasets</th>
<th>RF-BigData</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td>DOS.<em>versus</em>.normal</td>
<td>98</td>
</tr>
<tr>
<td>DOS.<em>versus</em>.PRB</td>
<td>100</td>
</tr>
<tr>
<td>DOS.<em>versus</em>.R2L</td>
<td>97</td>
</tr>
<tr>
<td>DOS.<em>versus</em>.U2R</td>
<td>93</td>
</tr>
<tr>
<td>normal.<em>versus</em>.PRB</td>
<td>94</td>
</tr>
<tr>
<td>normal.<em>versus</em>.R2L</td>
<td>92</td>
</tr>
<tr>
<td>normal.<em>versus</em>.U2R</td>
<td>93</td>
</tr>
</tbody>
</table>

**Cluster ATLAS: 16 nodes**
- Microprocessors: 2 x Intel E5-2620 (6 cores/12 threads, 2 GHz)
- RAM 64 GB DDR3 ECC 1600MHz
- Mahout version 0.8
Case of Study: (Global) Decision Trees for Big Data

- **Decision Trees in Spark**
  - **Differences** with respect to classical models
    - All the nodes in a level are learned with a single pass through the whole dataset
    - Numeric attributes are discretized into bins in order to reduce the computational cost

https://speakerdeck.com/jkbradley/mllib-decision-trees-at-sf-scala-baml-meetup
Global Decision Trees for Big Data

- **Decision Trees in Spark**
  - **Differences** with respect to classical models
    - All the nodes in a level are learned with a single pass through the whole dataset

![Decision Tree Diagram]
Global Decision Trees for Big Data

- **Decision Trees in Spark**
  - **Differences** with respect to classical models
    - Numeric attributes are discretized into bins in order to reduce the computational cost

![Diagram of Decision Trees](image)

`Continuous feature: x_j < (value)`

Bin 1
- # Not spam
- # Spam

Bin 2
- # Not spam
- # Spam

Bin 3
- # Not spam
- # Spam

Bin 4
- # Not spam
- # Spam

Bin 5
- # Not spam
- # Spam
Big Data Analytics: 2 books

9 cases of study

10 chapters giving a quick glance on Machine Learning with Spark
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Fuzzy-based models for Big Data learning

- OUTLINE
  - Data fragmentation and lack of data problem
  - Big Data classification with fuzzy models
  - Are FRBCSs robust with respect to the lack of data?
  - Conclusions and future challenges
Fuzzy-based models for Big Data learning

OUTLINE

- Data fragmentation and lack of data problem
- Big Data classification with fuzzy models
- Are FRBCSs robust with respect to the lack of data?
- Conclusions and future challenges
Rare cases or Small disjuncts are those disjuncts in the learned classifier that cover few training examples.


Big Data Classification: Data Fragmentation and Lack of Data Problem

It becomes very hard for the learning algorithm to obtain a model that is able to perform a good generalization when there is not enough data that represents the boundaries of the problem.
Big Data Classification: Data Fragmentation and Lack of Data Problem

**MapReduce**

Data Fragmentation with Parallel Processing + Model fusion

Small disjuncts arise with MapReduce data fragmentation. This problem is accentuated for imbalance classification: Lack of Data/lack of density between classes
The lack of data in the training data may cause the introduction of small disjuncts.

What it is also most significant, when the concentration of minority examples is so low that they can be simply treated as noise.
Lack of data

Left-C4.5, right-Backpropagation (Pima data set): These results show that the performance of classifiers is repaired as the training set size increases. This suggests that small disjuncts play a role in the performance loss of class imbalanced domains.

Big Data Classification: Data Fragmentation and Lack of Data Problem

Lack of data. Fuzzy models performance

Robustness to the lack of data?
Fuzzy-based models for Big Data learning

OUTLINE

- Data fragmentation and lack of data problem
- Big Data classification with fuzzy models
- Are FRBCSs robust with respect to the lack of data?
- Conclusions and future challenges
Uncertainty and Big Data

- Uncertainty is inherent to Big Data due to:
  - Heterogeneous sources
  - Variety in data
  - Incomplete data
  - Veracity in question

- Fuzzy Rule Based Classification Systems can manage:
  - Uncertainty
  - Vagueness
  - Lack of data/Data fragmentation
Linguistic Fuzzy Models as a robust solution for Big Data

- Linguistic FRBCSs have been used in distributed environments for a long time
- In the context of Big Data applications, MapReduce is the “key”:
  - simple,
  - fault-tolerant,
  - scalable.
- The programming framework differs from traditional schemes:
  - Map tasks imply the building of local / partial models
  - Reduce tasks aggregate partial models
Big Data Classification

with Fuzzy Models

Chi-FRBCS-BigData: A Case of Study

We choose a simple Learning Method to analyze the potential of FRBCSs for Big Data Classification

- MapReduce design based on the Chi et al. FRBCS algorithm (two different MapReduce processes)
  - Phase 1: Building the Fuzzy Rule Base
  - Phase 2: Estimating the class of samples belonging to Big Data sample sets
- Two versions that differ in the Reduce function of the building of the Fuzzy RB are considered
  - Chi-FRBCS-BigData-Max
  - Chi-FRBCS-BigData-Average

Chi-FRBCS

- Generates rules as “Rule R_j: IF x_1 IS A_1^j AND ... AND x_n IS A_n^j THEN Class = C_j with RW_j”
- Builds the fuzzy partition using equally distributed triangular membership functions
- Builds the RB creating a fuzzy rule associated to each example
- Rules with the same antecedent may be created:
  - Same consequent → Delete duplicated rules
  - Different consequent → Preserve highest weight rule

Z. Chi, H. Yan and T. Pham, Fuzzy algorithms with applications to image processing and pattern recognition, World Scientific, 1996.
Rule Base (No Weights):
If \( x_1 \) is \textit{small} and \( x_2 \) is \textit{small} then Class 2
Big Data Classification with Fuzzy Models

Rule Base (No Weights)
If $x_1$ is small and $x_2$ is small then Class 2
If $x_1$ is small and $x_2$ is medium then Class 2
Big Data Classification with Fuzzy Models

Rule Base (No weights)
If $x_1$ is small and $x_2$ is small then Class 2
If $x_1$ is small and $x_2$ is medium then Class 2
If $x_1$ is small and $x_2$ is large then Class 1
Big Data Classification with Fuzzy Models

Classification Boundaries

If $x_1$ is small and $x_2$ is small then Class 2
If $x_1$ is small and $x_2$ is medium then Class 2
If $x_1$ is small and $x_2$ is large then Class 1
... 
If $x_1$ is large and $x_2$ is large then Class 3

High interpretability
Standard rules (w/o weights) does not always get a good representation

**Rule Base (No Weights)**

If $x_1$ is *small* and $x_2$ is *small* then Class 2

If $x_1$ is *small* and $x_2$ is *medium* then Class 2

If $x_1$ is *small* and $x_2$ is *large* then Class 1

...  

If $x_1$ is *large* and $x_2$ is *large* then Class 3

**High Interpretability**  
**Low Accuracy**
Big Data Classification with Fuzzy Models

RB (No weights)
If $x_1$ is small and $x_2$ is medium then Class 2

RB (Weights == Certainty Factor)
If $x_1$ is small and $x_2$ is medium then Class 2 with 0.158
Building the RB with Chi-FRBCS-BigData: A Map Reduce approach

The key of a MapReduce data partitioning approach is usually on the reduce phase

Two alternative reducers (Max vs average weights)
Building the FRB with Chi-FRBCS-BigData-Max

R1: IF A1 = L1 AND A2 = L1 THEN C1; RW1 = 0.8743
R2: IF A1 = L2 AND A2 = L2 THEN C2; RW2 = 0.9142

RB1

R1: IF A1 = L1 AND A2 = L1 THEN C1; RW1 = 0.9254
R2: IF A1 = L1 AND A2 = L2 THEN C2; RW2 = 0.8842

RB2

R1: IF A1 = L2 AND A2 = L1 THEN C2; RW1 = 0.6534
R2: IF A1 = L1 AND A2 = L1 THEN C1; RW1 = 0.7142

RB3

R1: IF A1 = L1 AND A2 = L1 THEN C2; RW1 = 0.2143
R2: IF A1 = L3 AND A2 = L2 THEN C2; RW3 = 0.4715

RB4

R1: IF A1 = L2 AND A2 = L3 THEN C2; RW3 = 0.7784
R2: IF A1 = L1 AND A2 = L1 THEN C1; RW2 = 0.8215

RBn

REDUCE

R1: IF A1 = L1 AND A2 = L1 THEN C1; RW1 = 0.9254
R2: IF A1 = L2 AND A2 = L2 THEN C2; RW2 = 0.9142
R3: IF A1 = L1 AND A2 = L2 THEN C2; RW2 = 0.8842
R4: IF A1 = L2 AND A2 = L1 THEN C2; RW3 = 0.6534
R5: IF A1 = L3 AND A2 = L2 THEN C2; RW3 = 0.4715
R6: IF A1 = L2 AND A2 = L3 THEN C2; RW3 = 0.7784

RBn

Final RB generation

RB1, R2, C1, RW = 0.8743
RB2, R1, C2, RW = 0.9254
RB3, R2, C1, RW = 0.7142
RB4, R1, C2, RW = 0.2142
RB5, R2, C1, RW = 0.8215
Building the FRB with Chi-FRBCS-BigData-Ave

R₁: IF A₁ = L₁ AND A₂ = L₁ THEN C₁; RW₁ = 0.8743
R₂: IF A₁ = L₂ AND A₂ = L₂ THEN C₂; RW₂ = 0.9142
...

RB₁

R₁: IF A₁ = L₁ AND A₂ = L₁ THEN C₂; RW₃ = 0.9254
R₂: IF A₁ = L₁ AND A₂ = L₂ THEN C₂; RW₂ = 0.8842
...

RB₂

R₁: IF A₁ = L₂ AND A₂ = L₁ THEN C₂; RW₃ = 0.6534
R₂: IF A₁ = L₁ AND A₂ = L₁ THEN C₁; RW₁ = 0.7142
...

RB₃

R₁: IF A₁ = L₁ AND A₂ = L₁ THEN C₂; RW₁ = 0.2143
R₂: IF A₁ = L₃ AND A₂ = L₂ THEN C₂; RW₃ = 0.4715
...

RB₄

...

RBₙ

R₁: IF A₁ = L₂ AND A₂ = L₃ THEN C₂; RW₃ = 0.7784
R₂: IF A₁ = L₁ AND A₂ = L₁ THEN C₁; RW₂ = 0.8215
...

REDUCE

R₁: IF A₁ = L₁ AND A₂ = L₁ THEN C₁; RW₁ = 0.8033
R₂: IF A₁ = L₂ AND A₂ = L₂ THEN C₂; RW₂ = 0.9142
R₃: IF A₁ = L₁ AND A₂ = L₂ THEN C₂; RW₂ = 0.8842
R₄: IF A₁ = L₂ AND A₂ = L₁ THEN C₂; RW₃ = 0.6534
R₅: IF A₁ = L₃ AND A₂ = L₂ THEN C₂; RW₃ = 0.4715
R₆: IF A₁ = L₂ AND A₂ = L₃ THEN C₂; RW₃ = 0.7784
...

Final RB generation

RB₁, R₁, C₁, RW = 0.8743
RB₂, R₁, C₂, RW = 0.9254
RB₃, R₂, C₁, RW = 0.7142
RB₄, R₁, C₂, RW = 0.2143
RB₅, R₂, C₁, RW = 0.8215

RC₁, C₁, RWave = 0.8033
RC₂, C₂, RWave = 0.5699
Summary of Chi-FRBCS-BigData

Homeogenous fuzzy partitions shared by all Map tasks

Rule with equal antecedents are merged.

RWs are averaged

The original Chi fuzzy rule learning algorithm is applied

Consequent by P-CF

Big Data Classification with Fuzzy Models

Experimental Analysis: Chi-FRBCS-BigData

- 6 Datasets with two classes problem
- Stratified 10 fold cross-validation
- Parameters:
  - Conjunction Operator: Product T-norm
  - Rule Weight: Penalized Certainty Factor
  - Number of fuzzy labels per variable: 3 labels

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Ex.</th>
<th>#Atts.</th>
<th>Selected classes</th>
<th>#Samples per class</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLCP</td>
<td>5749132</td>
<td>2</td>
<td>(FALSE; TRUE)</td>
<td>(5728201; 20931)</td>
</tr>
<tr>
<td>Kddcup_DOS_vs_normal</td>
<td>4856151</td>
<td>41</td>
<td>(DOS; normal)</td>
<td>(3883370; 972781)</td>
</tr>
<tr>
<td>Poker_0_vs_1</td>
<td>946799</td>
<td>10</td>
<td>(0; 1)</td>
<td>(513702; 433097)</td>
</tr>
<tr>
<td>Covtype_2_vs_1</td>
<td>495141</td>
<td>54</td>
<td>(2; 1)</td>
<td>(283301; 211840)</td>
</tr>
<tr>
<td>Census</td>
<td>141544</td>
<td>41</td>
<td>(-_50000.; 50000+)</td>
<td>(133430; 8114)</td>
</tr>
<tr>
<td>Fars Fatal_Inj_vs_No_Inj</td>
<td>62123</td>
<td>29</td>
<td>(Fatal_Inj; No_Inj)</td>
<td>(42116; 20007)</td>
</tr>
</tbody>
</table>
## Big Data Classification with Fuzzy Models

### Analysis of the Performance, Precision

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Chi-FRBCS</th>
<th>8 maps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc$_{tr}$</td>
<td>Acc$_{ts}$</td>
</tr>
<tr>
<td>Poker_0_vs_1</td>
<td>63.72</td>
<td>61.77</td>
</tr>
<tr>
<td>Covtype_2_vs_1</td>
<td>74.65</td>
<td>74.57</td>
</tr>
<tr>
<td>Census</td>
<td>96.52</td>
<td>86.06</td>
</tr>
<tr>
<td>Fars_Fatal_Inj_vs_No_Inj</td>
<td>99.66</td>
<td>89.26</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>83.64</td>
<td>77.92</td>
</tr>
</tbody>
</table>

**Good precision!**
### Analysis of the Performance, Precision

<table>
<thead>
<tr>
<th>Datasets</th>
<th>16 mappers</th>
<th>32 mappers</th>
<th>64 mappers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chi-BigData-Max</td>
<td>Chi-BigData-Ave</td>
<td>Chi-BigData-Max</td>
</tr>
<tr>
<td>Poker_0_vs_1</td>
<td>62.18</td>
<td>59.88</td>
<td>62.58</td>
</tr>
<tr>
<td>Covtype_2_vs_1</td>
<td>74.77</td>
<td>74.72</td>
<td>74.77</td>
</tr>
<tr>
<td>Census</td>
<td>97.14</td>
<td>93.75</td>
<td>97.15</td>
</tr>
<tr>
<td>Fars_Fatal_Init_vs_No_Init</td>
<td>96.69</td>
<td>94.75</td>
<td>97.06</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>88.39</strong></td>
<td><strong>87.11</strong></td>
<td><strong>88.52</strong></td>
</tr>
</tbody>
</table>

- **Performance improves slightly with less maps** (alleviate the small sample size problem)
- **Chi-BigData-Ave obtains slightly better classification results**
Big Data Classification with Fuzzy Models

Analysis of the Performance, Number of rules

<table>
<thead>
<tr>
<th>Kddcup_DOS_vs_normal dataset</th>
<th>NumRules by map</th>
<th>Final numRules</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB₁ size: 211</td>
<td>RB₁ size: 211</td>
<td>RB₁ size: 301</td>
</tr>
<tr>
<td>RB₂ size: 212</td>
<td>RB₂ size: 212</td>
<td>RB₂ size: 212</td>
</tr>
<tr>
<td>RB₃ size: 221</td>
<td>RB₃ size: 221</td>
<td>RB₃ size: 221</td>
</tr>
<tr>
<td>RB₄ size: 216</td>
<td>RB₄ size: 216</td>
<td>RB₄ size: 216</td>
</tr>
<tr>
<td>RB₅ size: 213</td>
<td>RB₅ size: 213</td>
<td>RB₅ size: 213</td>
</tr>
<tr>
<td>RB₆ size: 210</td>
<td>RB₆ size: 210</td>
<td>RB₆ size: 210</td>
</tr>
<tr>
<td>RB₇ size: 211</td>
<td>RB₇ size: 211</td>
<td>RB₇ size: 211</td>
</tr>
<tr>
<td>RB₈ size: 214</td>
<td>RB₈ size: 214</td>
<td>RB₈ size: 214</td>
</tr>
</tbody>
</table>

Robustness to the lack of data, increasing the final number of rules

Class | Instance Number
normal | 972.781
DOS     | 3.883.370

![Graph showing accuracy test results](image)
Analysis of the Performance, Number of rules

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Chi-FRBCS Average NumRules</th>
<th>8 maps</th>
<th>Chi-BigData-Max Average NumRules</th>
<th>Chi-BigData-Ave Average NumRules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Census</td>
<td>31518.3</td>
<td>34278.0</td>
<td>34278.0</td>
<td></td>
</tr>
<tr>
<td>Covtype_2_vs_1</td>
<td>6962.7</td>
<td>7079.1</td>
<td>7079.1</td>
<td></td>
</tr>
<tr>
<td>Fars_Fatal_Inj_vs_No_Inj</td>
<td>16843.3</td>
<td>17114.9</td>
<td>17114.9</td>
<td></td>
</tr>
<tr>
<td>Poker_0_vs_1</td>
<td>51265.4</td>
<td>52798.1</td>
<td>52798.1</td>
<td></td>
</tr>
</tbody>
</table>

Robustness to the lack of data, increasing the final number of rules

This may cause an improvement in the performance!!
Analysis of the Performance, Runtime

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Chi-FRBCS Runtime (s)</th>
<th>Chi-BigData-Max Runtime (s)</th>
<th>Chi-BigData-Ave Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Census</td>
<td>38655.60</td>
<td>1102.45</td>
<td>1343.92</td>
</tr>
<tr>
<td>Covtype_2_vs_1</td>
<td>86247.70</td>
<td>2482.09</td>
<td>2512.16</td>
</tr>
<tr>
<td>Fars_Fatal_Inj_vs_No_Inj</td>
<td>8056.60</td>
<td>241.96</td>
<td>311.95</td>
</tr>
<tr>
<td>Poker_0_vs_1</td>
<td>114355.80</td>
<td>5672.80</td>
<td>7682.19</td>
</tr>
<tr>
<td>Average</td>
<td>61828.93</td>
<td>2374.82</td>
<td>2962.56</td>
</tr>
</tbody>
</table>

KddCUP’99

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>972.781</td>
</tr>
<tr>
<td>DOS</td>
<td>3.883.370</td>
</tr>
</tbody>
</table>

Maps number | Seconds
---|----------
8   | 116.218,26
16  | 29.820,01
32  | 7.708,96
64  | 2.096,34
132 | 1.579,77

Performance improves slightly with less maps (alleviate the small sample size problem)
Fuzzy-based models for Big Data learning

OUTLINE

- Data fragmentation and lack of data problem
- Big Data classification with fuzzy models
- Are FRBCSs robust with respect to the lack of data?
- Conclusions and future challenges
Experimental Framework

Two versions of Chi algorithm will be used:

- **Chi-FRBCS-BigData** (noted as \textbf{ChiBD}): MapReduce implementation
- **Sequential Chi** (noted as \textbf{ChiStd}): Results of 1Map (from 1 to 128)
Are FRBCS robust with respect to the lack of data?

**Accuracy** obtained by Chi-FRBCSBigData (ChiBD) and a sequential run of Chi (ChiStd) with respect to #Maps

Good precision
Are FRBCS robust with respect to the lack of data?

Analyzing the capabilities of representation of fuzzy models:

**Variation of accuracy (%)** 1 Map (100% data) from 8 to 128 Maps.

**Algorithms:** Chi-FRBCS-BigData (ChiBD), Chi (ChiStd) and C4.5

1. Advantage of aggregating the local models in MapReduce.
2. Fuzzy-Chi better than C4.5:
   - Few examples derive rules that represent larger regions.
   - How to provide a precise computation of the RWs.
Analysis for the number of rules within Maps

- Higher # maps, the smaller the contribution of each RBi.
- Better interpretability within each local fuzzy classifier,
- Few fuzzy rules: good representation of the problem space
- Global RB composed by more rules (RB_i aggregation)
Analysis for the number of rules within Maps

• **More Maps \(\Rightarrow\) More Rules.**
• P-CF RW: **Filter** rules (RW < 0) from local RBs.
• Sparsely distributed data: values of RWs become higher.
Fuzzy rules in Reduce:
Same antecedent

- High number of co-occurrences, especially for a low #maps:
  - The larger the volume of input data, more similar clusters are expected.
  - Same fuzzy labels chosen from these clusters

- Equal rules found among maps: higher density of data within the problem space.
Fuzzy rules in Reduce: Double consequent

- How many rules in conflict arrive to a Reduce task.
- Rules related to overlapped regions.
- Low percentage:
  - Probably found within a single map process, and discarded due to the RW computation.
Fuzzy rules in Reducer: Unique rules

- Rules generated in a single Map task.
- This value is practically the same for all Map case studies.
- Instances far from areas of high density of data.
- Their influence is independent of the data distribution.
Contribution of fuzzy rules to classification

“Winning rule” inference: “strongest” rule determines class label

Types of rules analyzed:

• All rules without distinction (noted as “Total”),
• Rules whose antecedents have been mined along different maps (noted as \([R>1]\))], and
• Rules that are generated on a single map (\([R = 1]\)).
Contribution of fuzzy rules to classification

“Winning rule” inference: “strongest” rule determines class label

- Not all rules are finally used for classification.
- Strong gap between “repeated” and “unique” rules
- The degree of contribution of “unique” rules is low.
Fuzzy-based models for Big Data learning

OUTLINE

- Data fragmentation and lack of data problem
- Big Data classification with fuzzy models
- Are FRBCSs robust with respect to the lack of data?
- Conclusions and future challenges
Concluding Remarks

- **Linguistic fuzzy models for Big Data** under the MapReduce framework:
  - Manages big datasets
  - Without damaging the classification accuracy
  - Fast response times (increasing with the number of Maps)

- **Robustness** of fuzzy models when addressing the problem of **small disjuncts** and **lack of data**:
  - High degree of *replicated* rules along maps: partial RBs represent accurately a high percentage of the problem.
  - *Local models* learned in each Map task are of *high quality*. Aggregation boosts the recognition ability of the FRBCS.
  - Those fuzzy rules that are obtained from more than a map *contribute the most* to the final classification
Future Challenges

- Designing new learning methodologies for FRBCS in MapReduce:
  - Reduce phase for approximate fuzzy models
  - Deep analysis “ensembles vs. fusion of rules”
  - Analysis on small disjuncts preprocessing for fuzzy models
  - New fuzzy models based on accurate algorithms
  - Study the effect of \textit{RW computation} in the quality of the fuzzy model
  - Application of good practices by means of scalable models:
    - Contextualization of the Data Base
    - Optimization of the Rule Base
- A promising line of work for the design of high performance Fuzzy Models for Big Data
Big Data Classification with Fuzzy Models

Code for our approaches:  https://github.com/aFdezHilario

Fuzzy Rule Based System for classification (w/wo cost sensitive)

Evolutionary Fuzzy System for Rule Selection
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
“Without Analytics, Big Data is just Noise”

Guest post by Eric Schwartzman, founder and CEO of Comply Socially

http://www.briansolis.com/2013/04/without-analytics-big-data-is-just-noise/

Hal Varian
Google Chief Economist
What are the challenges of analyzing Big Data?

Big data are characterized by high dimensionality and large sample size. These two features raise three unique challenges: (i) high dimensionality brings noise accumulation, spurious correlations and incidental homogeneity, (ii) high dimensionality combined with large sample size creates heavy computational cost and algorithmic instability, (iii) the massive samples in Big Data are typically aggregated from multiple sources at different time points using different technologies. This creates issues of heterogeneity, experimental variations and statistical biases, and requires us to develop more adaptive and robust procedures.
RECALL: Big data as a concept is defined around five aspects:

- Data volume,
- Data velocity,
- Data variety,
- Data veracity and
- Data value.
a) While the **volume, variety and velocity** aspects refer to data generation process and how to capture and store the data,

b) **Veracity and value** aspects deal with the **quality and the usefulness of the data** leading to the point.
Smart Data

Smart Data (veracity and value) aims to filter out the noise and hold the valuable data, which can be effectively used by enterprises and governments for planning, operation, monitoring, control, and intelligent decision making.

What makes data smart?

Three key attributes for data to be smart, it must be **accurate**, **actionable**, and **agile**.
The key is to explore how Big Data can become Smart Data.

“Without Analytics, Big Data is just Noise”
Eric Schwartzman

Big Data Preprocessing + Analytics = Smart Data

“Here’s a list of 100,000 warehouses full of data. I’d like you to condense them down to one meaningful warehouse.”
Big Data (Analytics) → Smart Data

Smart Data

Quality data for quality decisions!

Big Data

Data Science

Data Preprocessing

Model building Predictive and descriptive Analytics
What is included in data preprocessing?

- Data Cleaning
- Data Normalization
- Data Transformation
- Missing Values Imputation
- Data Integration
- Noise Identification
- Feature Selection
- Instance Selection
- Discretization

Fig. 1.3 Forms of data preparation

Fig. 1.4 Forms of data reduction
Big data preprocessing also must spend a very important part of the total time in a big data analytic process.
Big Data at SCI$^2$S - UGR

Bird's eye view SCI$^2$S website

http://sci2s.ugr.es/BigData

Big Data: Algorithms for Data Preprocessing, Computational Intelligence, and Imbalanced Classes

The web is organized according to the following summary:

1. Introduction to Big Data
2. Big Data Technologies: Hadoop ecosystem and Spark
3. Big Data preprocessing
4. Imbalanced Big Data classification
5. Big Data classification with fuzzy models
6. Classification Algorithms: k-NN
7. Big Data Applications
8. Dataset Repository
9. Literature review: surveys and overviews
10. Keynote slides
11. Links of interest

Big Data Preprocessing

1. Introduction to Data Preprocessing
2. Feature Selection
3. Feature Weighting
4. Discretization
5. Prototype Generation
Feature Selection

Fast-mRMR: an optimal implementation of minimum Redundancy Maximum Relevance algorithm

https://github.com/sramirez/fast-mRMR

This is big data based implementation of the classical feature selection method: minimum Redundancy and Maximum Relevance (mRMR); (Hanchuan Peng, Fuhui Long, and Chris Ding "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp.1226-1238, 2005).

This includes several optimizations such as: cache marginal probabilities, accumulation of redundancy (greedy approach) and a data-access by columns.
Feature Selection

Fast-mRMR: Minimum Redundancy Maximum Relevance algorithm

**TABLE IV: Selection time by dataset and threshold (in seconds)**

<table>
<thead>
<tr>
<th># Features</th>
<th>kdddb</th>
<th>url</th>
<th>dna</th>
<th>ECBDL14</th>
<th>epsilon</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>283.61</td>
<td>94.06</td>
<td>97.83</td>
<td>332.90</td>
<td>111.42</td>
</tr>
<tr>
<td>25</td>
<td>774.43</td>
<td>186.22</td>
<td>148.78</td>
<td>596.31</td>
<td>173.39</td>
</tr>
<tr>
<td>50</td>
<td>1365.82</td>
<td>333.70</td>
<td>411.84</td>
<td>1084.58</td>
<td>292.07</td>
</tr>
<tr>
<td>100</td>
<td>2789.55</td>
<td>660.48</td>
<td>828.35</td>
<td>2420.94</td>
<td>542.05</td>
</tr>
</tbody>
</table>
From high dimensionality to small dimensionality:

Eases the application of fuzzy learning systems

**TABLE IV: Selection time by dataset and threshold (in seconds)**

<table>
<thead>
<tr>
<th># Features</th>
<th>kdddb</th>
<th>url</th>
<th>dna</th>
<th>ECBDL14</th>
<th>epsilon</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>283.61</td>
<td>94.06</td>
<td>97.83</td>
<td>332.90</td>
<td>111.42</td>
</tr>
<tr>
<td>25</td>
<td>774.43</td>
<td>186.22</td>
<td>148.78</td>
<td>596.31</td>
<td>173.39</td>
</tr>
<tr>
<td>50</td>
<td>1365.82</td>
<td>333.70</td>
<td>411.84</td>
<td>1084.58</td>
<td>292.07</td>
</tr>
<tr>
<td>100</td>
<td>2789.55</td>
<td>660.48</td>
<td>828.35</td>
<td>2420.94</td>
<td>542.05</td>
</tr>
</tbody>
</table>
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Beyond interpretability vs. accuracy trade-off

- Fuzzy systems began to be used for expert knowledge representation, knowledge linguistically expressed that allowed the development of fuzzy logic controllers.

- Later they were used to represent knowledge extracted database and began the rise of automatic learning of fuzzy systems, being developed multiple proposals with different tools such as neural networks, genetic algorithms, clustering, ad-hoc models, ...

- Then we witness the long and rich debate about the balance between interpretability and precision fuzzy modeling, and interpretability role played in the design and subsequent use of fuzzy systems.
Beyond interpretability vs. accuracy trade-off

Learning from data  25 years studies  1992-2016

We are caught in a spiral
Interpretability vs Precision

Debate as a dialectical exercise that lasts and that in most cases does not end with conclusions that have real value at a practical level.

The horizon is lost and we have not progressed toward open problems.
Several questions can be raised to reflect on the interpretability:

- When is useful the linguistic description or the fuzzy rules interpreted with $\alpha$-cuts?
- Why must we bet on the use of models of fuzzy rules models such as divide and conquer decision trees or crisp rule based systems?
- Are useful interpretable models for complex problems with lots of data?
- Should we continue to design more complex and not interpretable fuzzy models via models fusion as ensembles, to be competitive in precision?
Beyond interpretability vs. accuracy trade-off

Suggestion to design “fuzzy intelligent systems” from data

Designing “Intelligent Systems” from data

What kind of systems do we want?
➢ They should perform well
➢ They should be understandable (not a black box)
➢ They should be simple (why!!)

"Make everything as simple as possible, but not simpler."

“I am not a genius, I am just curious. I ask many questions, and when the answer is simple, then God is answering.” (Albert Einstein)

“Nature is pleased with simplicity. And nature is no dummy” (Isaac Newton)

performing well, understandable and simple
Beyond interpretability vs. accuracy trade-off

There are other models beyond rule-based models (performing well, understandable and simple).

An interesting model: fuzzy prototypes represented by nearest neighbor techniques.

It deserves to go back over this model and to extend his use in data science/big data.
There are other models beyond rule-based models.

Using the similarity among data: Rough Sets Theory (*Lower approximation and positive region vs Upper approximation and negative region*)

\[
Ry = \{ x \mid x \in X \text{ and } (x, y) \in R \}
\]

\[
y \in R\downarrow A \iff [y]_R \subseteq A
\]
\[
y \in R\uparrow A \iff [y]_R \cap A \neq \emptyset
\]

\[
y \in R\downarrow A \iff (\forall x \in X)((x, y) \in R \Rightarrow x \in A)
\]
\[
y \in R\uparrow A \iff (\exists x \in X)((x, y) \in R \land x \in A)
\]
Beyond interpretability vs. accuracy trade-off

Using the similarity among data: Fuzzy Rough Sets Theory

\[
R_y(x) = R(x, y) \\
R \downarrow A(y) = \inf_{x \in X} T(R(x, y), A(x)) \\
R \uparrow A(y) = \sup_{x \in X} T(R(x, y), A(x))
\]

**State of the art**
S Vluymans, L D’eer, Y Saeys, C Cornelis
*Applications of Fuzzy Rough Set Theory in Machine Learning: a Survey*
Fundamenta Informaticae 142 (1-4), 53-86, 2015

**An application for imbalanced data sets – State of the art**
IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification
E Ramentol, S Vluymans, N Verbiest, Y Caballero, R Bello, C Cornelis, F. Herrera
Beyond interpretability vs. accuracy trade-off

They are additional fuzzy models in the literature, whose use is not analyzed at a theoretical or practical level in depth: Fuzzy decision trees, fuzzy SVM, ...

Today it is accepted that fuzzy systems are a good tool in data modeling, although not yet have developed studies analyzing the role "when and where they should be used and why"

This is a challenge beyond interpretability-accuracy debate!!.
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Fuzzy Big Data Science: Opportunities

- The most effective classification algorithms do not allow to interpret the models.
- The interpretation of the models is an essential feature of rule-based systems (RBSs).
- There are applied areas where experts want to know why decisions.
- RBSs fill a very important gap in Data Science.

Fuzzy Rule Based Systems play an important role as RBSs for the description of the facts in terms of knowledge represented by rules.
Fuzzy Big Data Science: Opportunities

There are open multiple scenarios of data science and their challenges and opportunities for fuzzy modeling:

- Multi-Label
- Multi-Instance Learning
- Multiview Learning
- Semi-supervised Learning
- Transfer Learning
- Big Data
Fuzzy Big Data Science: Opportunities

There are open multiple scenarios of data science and their challenges and opportunities for fuzzy modeling:
Fuzzy Big Data Science: Opportunities

There are open multiple scenarios of data science and their challenges and opportunities for fuzzy modeling:

- Multi Label
- Multi Instance Learning
- Multi view Learning

Tweets with image and text

Even a packed Edinburgh to Glasgow evening ...

For our first time at state we are ecstatic ...

Feature Extraction

- Textual Feature (BoW)
- Visual Feature (SIFT, GIST, Sentibank, etc.)
- Multi-view Feature (Multi-model DBM)

Model Learning

- Linear SVM

Sentiment Prediction

gecanAslan murder: men kept at bay at the funeral. Women of #Turkey are furious as hell.
Fuzzy Big Data Science: Opportunities

There are open multiple scenarios of data science and their challenges and opportunities for fuzzy modeling:

- Multi Label
- Multi Instance Learning
- Multiview Learning
- Semi-supervised Learning
- Transfer Learning
- Big Data
Outline

- Introduction to Big Data
- Big data analytics
- Fuzzy-based models for Big Data Learning
- Smart Data: The missing bridge between Big Data and real applications to get high quality data
- Fuzzy Data Science: Beyond Interpretability-Accuracy Tradeoff
- Fuzzy Big Data Science: Opportunities
- Final Comments
Final Comments

Data Mining, Machine learning and data preprocessing: Huge collection of algorithms

Big Data Analytics

Big Data: A small subset of algorithms

Big Data Preprocessing: A few methods for preprocessing in Big Data analytics. Evolutionary models are a promising approach.

Soft computing approaches are useful to tackle big data analytics
Fuzzy models for big data: Robustness to the lack of data for the data fragmentation, producing high performance (accuracy).

The focus should be on
- a) designing models for maps and
- b) the combination phase (reduce).

Potential applications in multiple scenarios:
“We have to place the Fuzzy Systems on the stage where they can play an important role in Data Science”.

**Deep Learning:** Transforms information to provide knowledge in terms of decisions, but do not explain why decisions (resurface neural networks).

**Fuzzy Rule Based Systems/Fuzzy Models:** Interpret information to provide knowledge in the form of linguistic rules or understandable and simple fuzzy models. They should have a greater presence in the Data Science forums (performing well, understandable and simple).
Final Comments: Questions for discussion

- **Interpretability.** Are we caught in a spiral? Why more discussions on interpretability? What are the limits?

- **Beyond classical classification and regressions.** What are the possibilities of fuzzy systems in the new data science areas?

- **Big Data.** Why to design fuzzy systems for big data?

- **Deep learning in the center of the Artificial Intelligence. Deep Learning vs fuzzy systems.** Why to use fuzzy systems? What is the position of Fuzzy logic in the Artificial Intelligence board? “Designing Intelligent Fuzzy Systems performing well, understandable and simple”
Big data and analytics: a large challenge offering great opportunities

- A small subset of algorithms.
  It is necessary to redesign new algorithms

- Computing Model
  - Accuracy and Approximation
  - Efficiency requirements for Algorithm

- Big Data Preprocessing
  - Noise in data distorts
    Need automatic methods for “cleaning” the data

- Missing values management
- Big Data Reduction

- Quality data for quality models in big data analytics

Final Comments
Quality decisions must be based on quality big data and quality models.

Computational Intelligence may be very useful to create new high quality big data analytics models.
Big Data: Algorithms for Data Preprocessing, Computational Intelligence, and Imbalanced Classes

The web is organized according to the following summary:

1. Introduction to Big Data
2. Big Data Technologies: Hadoop ecosystem and Spark
3. Big Data preprocessing
4. Imbalanced Big Data classification
5. Big Data classification with fuzzy models
6. Classification Algorithms: k-NN
7. Big Data Applications
8. Dataset Repository
9. Literature review: surveys and overviews
10. Keynote slides
11. Links of interest

http://sci2s.ugr.es/BigData
Thanks!!!

Fuzzy Models for Data Science and Big Data
Fuzzy Models for Data Science and Big Data

Francisco Herrera and Alberto Fernández
Soft Computing and Information Intelligent Systems (SCI²S)
University of Granada, Spain
Email: {herrera, alberto}@decsai.ugr.es
http://sci2s.ugr.es

Isaac Triguero
School of Computer Science
Automated Scheduling Optimisation and Planning (ASAP)
University of Nottingham
United Kingdom
Isaac.Triguero@nottingham.ac.uk