Massive scale analytics with Stratosphere using R

Jose Luis Lopez Pino
jllopezpino@gmail.com

Database Systems and Information Management
Technische Universität Berlin

Supervised by Volker Markl

Advised by Marcus Leich, Kostas Tzoumas

August 28, 2014
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Motivation</th>
<th>Our approach</th>
<th>Related work</th>
<th>Conclusions and Future Work</th>
</tr>
</thead>
</table>

**Introduction**
Data analysis to the masses

- Deep analytics\(^1\): sophisticated statistical methods like linear models, clustering or classification that frequently are used to extract knowledge from the data.
  - Data warehousing and BI can’t answer all the questions.
  - The ever-growing number of new data sources and tools make it worse.
- There is demand for this questions.
- In small scale: data pipelining tools (RapidMiner) and numerical computing environments (R, Matlab or SPSS).
- Big data brings new opportunities to the market but also presents unfamiliar challenges.

Options

- **R:**
  - R is a numerical computing environment and DSL for stats.
  - Not a query language unlike SQL.
  - Successful for small scale (in combination with CRAN packages).

- **MapReduce/Hadoop:**
  - Highly parallel programs but lack of expressivity.
  - HDFS: a de-facto standard to store big amounts of data.

- **Stratosphere:**
  - Platform for massively parallel computing / big data analytics.
  - PACT: MapReduce + New operators + Iterations.
Basic terms and definitions

- **KDD** is compound of nine steps: understanding the domain and the goals, creating the target source, cleaning and processing the source, data reduction and projection, choosing a data mining method, choosing the data mining algorithm, mining the data, interpretation of the patterns.

Figure: Overview of the process

---

Motivation
Clustering

Algorithm 1: Clustering with K-Means

Input: inputFilePath, outputFilePath

1 Read dataPoints from inputFilePath.
2 clusters ← cluster dataPoints using different number of clusters.
3 dataSample ← take a sample of dataPoints.
4 d ← Euclidean distance matrix of dataSample.
5 dunn ← Compute the Dunn indices of clusters using d.
6 Plot the different Dunn indices as a bar chart.
7 best ← choose the solution with maximum Dunn index.
8 Plot dataSample visualizing each cluster of best in a different colour.
9 outputFilePath ← append best to dataPoints.
Algorithm 2: Classification with Naive Bayes

**Input:** inputTraining, inputUnlabelled

**Output:** predictions

1. Read $trainingInstances$ from $inputTraining$.
2. Read $unlabelledInstances$ from $inputUnlabelled$.
3. $model \leftarrow$ create a Naive Bayes model based on $trainingInstances$.
4. $predictions \leftarrow$ predict the classes for $unlabelledInstances$ using $model$. 
## Frequent Terms

### Algorithm 3: Most frequent terms

**Input:** inputFile

1. `allWords ← Read all the words from inputFile.`
2. `interestingWords ← Remove all the stop words from allWords.`
3. `frequentWords ← Choose the most frequent words in interestingWords.`
4. **foreach** word in frequentWords **do**
   5. `synonyms ← Find all the synonyms of word.`
   6. Find the intersection between synonyms and words
Writing massively parallel programs

- It is a cumbersome and onerous process.
- We need of single tools.
- We need tools that can process from a small amount of data up to very large volumes.
- The majority of data researchers are strongly skilled in R and statistics and poorly skills in Big Data systems and implementation of machine learning algorithm.\(^3\) \(^4\)
- Although Stratosphere offers a more expressive interface, writing a parallel program is still not a trivial job.

\(^3\)Harlan Harris, Sean Murphy, and Marck Vaisman. *Analyzing the Analyzers: An Introspective Survey of Data Scientists and Their Work*. O’Reilly Media, Inc., 2013

Relation with the KDD process

- Data extraction is covered by other solutions.
- Pre-processing and transformation seem difficult.
- Data mining: where we have a competitive advantage.
- Data visualization is a different problem.
Design goals

- Easiness: ready-to-use algorithms.
- Design a library.
- Facilitate working with data.
- Easy to distribute.
- Focus on algorithms that scale.
Our approach
Architecture

- **Libraries**
  - Machine Learning
  - Regression models
  - Matrix transformations
  - Common statistical measures

- **Client (R)**
  - R package
  - Distributed algo.
  - Job execution
  - File manipulation

- **Stratosphere**
  - PACT-Client
  - JobManager
  - HDFS

  JAR file → Client (R) → R package

  JAR file + user parameters → PACT-Client → JobManager

  Files

Contact Information:
Jose Luis Lopez Pino jlopezpino@gmail.com
Architecture
Library: Goals

- Classification, clustering and regression.
- *No Free Lunch Theorem*: more than one algorithm.
- Presence in other ML libraries.
- Large-scale.
- Ensemble scenarios.
# Library: Example

Table 1: Classification algorithms in different machine learning libraries

<table>
<thead>
<tr>
<th>Technique</th>
<th>R</th>
<th>scikit</th>
<th>Mahout</th>
<th>MADlib</th>
<th>MLbase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>glm</td>
<td>Y</td>
<td>Y†</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>CORElearn</td>
<td>Y</td>
<td>Y</td>
<td>Y†</td>
<td>N</td>
</tr>
<tr>
<td>Perceptron</td>
<td>rminer</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SVM</td>
<td>e1071</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Quadratic classifiers</td>
<td>DiscrMiner</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>K-Nearest neighbor</td>
<td>CORElearn</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Boosting</td>
<td>gbm</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Random forests</td>
<td>randomForest</td>
<td>Y</td>
<td>Y</td>
<td>Y†</td>
<td>N</td>
</tr>
<tr>
<td>Neural network</td>
<td>nnet</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Gene Expression</td>
<td>GeneReg</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Bayesian networks</td>
<td>BayesTree</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Hidden Markov models</td>
<td>HiddenMarkov</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Learning vector quantization</td>
<td>LVQTools</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
R package

- Easy to distribute.
- Organized in namespaces.
- Submitting jobs to the cluster.
- Working with files.
- Mining.
- Configuration.
Example: Code

```r
# Required library
install.packages('e1071')
library(e1071)

# Load the required datasets
set_train <- read.csv("train")
set_classify <- read.csv("unlabelled")

# Create the model
model <- naiveBayes(set_train)

# Predict the class
p <- predict(model, set_classify)

# Write to disk
result <- data.frame(set_classify, p)
write.csv(result, "output")

# Required libraries
install.packages("stratosphereR")
library(stratosphereR)
library(e1071)
library(pmml)

# Load the required dataset
set_train <- stratosphere.file.toDataFrame("/train")

# Create the model
model <- naiveBayes(set_train)

# Output the PMML representation
pmmlf <- pmml(model, predictedField="Species")

# Classify the unlabelled data using the PMML model
stratosphere.mining.classify("unlabelled",
        "output",
        pmmlf)
```
Example: Non-parallel classification example

R client → Main memory → File system

Labelled instances → NB trainer → model → Unlabelled instances → NB classification

Classified instances → Classified instances
Example: Parallel classification example

- **R client**
- **Main memory**
- **Cluster execution**
- **Distributed file system**

**Flowchart:***
- **NB trainer** produces **model**.
- **HDFS service** processes **labelled instances**.
- **Stratosphere program** handles **unlabelled instances**.
- **Distributed file system** distributes files.
- **Cluster execution** runs tasks.
Example: Parallel clustering example
Performance

- Competitive and even faster than native R programs thanks to the pipelining for every parallelizable programs in the same (small) file size range.
- Competitive with R for data mining tasks with a lot of iterations in the same file size range.
- Able to process files of a volume that is inaccessible for R.
- Able to scale to gigabyte level without significant loss.
Performance: Frequent Terms example
Performance: Most favorable case to R

![Figure: KMeans 100 iterations](image-url)
Performance: Breakdown example

Figure: Clustering example nonparallel breakdown (Time in seconds)
Performance: Scalability example

**Figure:** Frequent Terms parallel scalability
Related work
Data mining libraries

▶ Don’t scale: Weka and sci-kit.
▶ Large-scale:
  ■ Mahout: limited set of problems.
  ■ MLlib: also facilitates implementation of new algorithms.
  ■ Oryx.
▶ In-database: MADlib and PivotalR.
Data intensive computation with R

- External memory.
  - Don’t scale-out: biglm, bigmemory, ff, foreach.
  - RevoScaleR: xdf files and Hadoop.
- Divide and recombine: it’s necessary to use the MR model.
- Query languages:
  - Limited expressivity.
  - Good for the first step of the KDD process.
- Distributed collection manipulation:
  - Limited set of operators.
  - Presto and SparkR.
Conclusions and Future Work
Conclusion

- **Contributions:**
  - Library definition.
  - File manipulation and cluster interaction.
  - Scenarios that proof the concept.

- Code very similar to the original one.
- Promising performance evaluation.
Future work

- Improvements in the library.
- Hybrid approaches.
- Distributed evaluation.
- Improvements in the architecture.
Essential bibliography


Recap

1. Introduction
   - Data analysis to the masses
   - Options
   - Basic terms and definitions

2. Motivation
   - Motivating problems
   - Writing massively parallel programs
   - Relation with the KDD process
   - Design goals

3. Our approach

4. Related work
   - Data mining libraries
   - Data intensive computation with R

5. Conclusions and Future Work
   - Conclusion
   - Future work
   - Essential bibliography