Alok Singh
IBM Spark Technology Center

R4ML: An R Based Scalable Machine Learning Framework
Alok Singh

Principal Engineer at the IBM Spark Technology Center. He is leading the R4ML team at IBM. He has built/architected analytical frameworks and implemented various machine learning algorithms.


Contact:
email- singh_alok@hotmail.com
Github -https://github.com/aloknsingh
IBM Spark Technology Center

Founded in 2015.
Location:
Physical: 505 Howard St., San Francisco CA
Web: [http://spark.tc](http://spark.tc)  Twitter: [@apachespark_tc](http://twitter.com/apachespark_tc)

Mission:
Contribute intellectual and technical capital to the Apache Spark community.
Make the core technology enterprise- and cloud-ready.
Build data science skills to drive intelligence into business applications — [http://bigdatauniversity.com](http://bigdatauniversity.com)

Key statistics:
About 50 developers, co-located with 25 IBM designers.
Major contributions to Apache Spark [http://jiras.spark.tc](http://jiras.spark.tc)
Apache SystemML is now an Apache Incubator project.
Founding member of UC Berkeley AMPLab and RISE Lab
Member of R Consortium and Scala Center
Introducing R4ML
R4ML is an R-language front end to Apache SystemML that integrates seamlessly with Apache Spark’s SparkR APIs.
SparkR is a Thin Wrapper Over Apache Spark

Brings in the powerful statistics from R to Spark

Source: SparkR talk by Shivaram Venkatraman in Spark Summit 15
Linear Algebra is the Language of Machine Learning.

Linear algebra is powerful, precise, and high-level.

Express complex transformations over large arrays of data... in a clear and unambiguous way... using a small number of operations.
Linear Algebra at Scale via Distributed Computing

- In R Linear Algebra and various matrix packages are very useful to create the new algorithms.

- Wouldn’t it be nice that all the linear algebra scale to millions and billions of rows?

- Wouldn’t it be nicer if user can write his code without worrying about the internal distributed framework.
Apache SystemML (incubating)  
Top-level project!

Goal: Take the linear algebra formulation of an algorithm and make it scale automatically

IBM Research technology, now an open-source Apache top level project.
To learn more: http://systemml.apache.org
L₂ regularized linear regression by the conjugate gradient method in DML:

```r
X = read.csv(...);  # n x m feature matrix
y = read.csv(...);  # n x 1 feature vector
maxi = 50; lambda = 0.001; ...

r = -(t(X) %*% y)
norm_r2 = sum(r * r); p = -r  # initial gradient
w = matrix(0, ncol(X), 1); i = 0
while(i < maxi & norm_r2 > norm_r2_trgt) {
    q = ((t(X)%*%(X%*%p) + lambda*p);  # compute conjugate gradient
    alpha = norm_r2 / sum(p * q)  # compute step size
    # update model and residuals
    w = w + alpha*p; r = r + alpha*q
    old_norm_r2 = norm_r2
    norm_r2 = sum(r^2); i = i + 1
    p = -r + norm_r2/old_norm_r2 * p
}
write.csv(w, ...);
```

Variables are matrices

Expressions use linear algebra operations
R4ML Details…
- Motivation
- Architecture
R4ML Architecture

- CRAN package
- R4ML adaptor pattern on top of sparkR APIs and systemML
  - R4ml.frame and r4ml.matrix
  - ML Algos
  - MLCtxt
- Spark
  - R workers
  - Distributed FS
- Apache SystemML

LEGENDS
- Network Comm
- InMem or InProcess Comm
How R4ML helps with scalable Machine Learning
A Typical ML Flow

- Read input data
- Pre-process
- Run built-in ML algorithm
- Run a custom ML algorithm
- Scoring
```r
library(R4ML) # load the library
r4ml.session() # init the session

hdfs_or_local_file = "sample_file.csv" # local or hdfs file
r4f <- r4ml.read.csv(hdfs_or_local_file, header = TRUE, sep = ",")
```
Pre-processing

• Typical machine learning use cases require many pre-processing steps

• R4ML provides and enhances these commonly used pre-processing options:
  • NA removal
  • Binning
  • Normalizing
  • Recoding
  • One-hot encoding

• R4ML also supports all the standard SparkR pre-processors
Pre-processing Example

```r
r4f.tx <- r4ml.ml.preprocess(# run preprocessing
r4f,
dummycodeAttrs = "Species", # one hot encoding
recodeAttrs=c("Species"), # recoding
scalingAttrs=c("Petal_Length"), # scaling and normalization
binningAttrs = c("Sepal_Length", "Sepal_Width"), numBins=4, # binning
missingAttrs = c("Petal_Length", "Sepal_Width"), # na handling
imputationMethod = c("global_mean", "constant"),
imputationValues = list("Sepal_Width" = 40), omit.na="Petal_Width")

r4_mat <- as.r4ml.matrix(r4f.tx$data) # the matrix after transform
r4_mat_info <- r4f_tx$metadata # contains the meta info about transform
```
R4ML’s Built-in ML Algorithms

- **Regression**
  - Linear Model (LM)
  - Generalized Linear Model (GLM)
  - Step Linear Model (Step.LM)
- **Classification**
  - Multi logistic Regression (MLOGIT)
  - Support Vector Machine (SVM)
- **Factorization**
  - Principal Component Analysis (PCA)
  - Alternating Least Square (ALS)
- **Survival analysis**
  - Kaplan Meier (KM) Survival model
  - Cox Proportional Hazard model
Training & Scoring GLM Example

```r
# split the data into training and test (continue from previous examples)
split_r4_mat <- r4ml.sample(r4_mat, perc=c(0.8,0.2))
train <- split_r4_mat[[1]]
test <- split_r4_mat[[2]]

glm_m <- r4ml.glm (Petal_Length ~ ., data = train, lambda = 1.1)

glm_pred <- predict(glm_m, test)

# statistics
glm_pred$statistics  # user can see R-squared, dispersion etc
r4ml.glm$predictions  # prediction columns
```
Custom ML algorithms

- Allows user to create the custom machine learning algorithms for POC.
- Works via the Apache SystemML DML connector.
- User need to attach the input and output variables from R/R4ML to the DML variables.
- User can create the library using the R package management.
# Let's create a simple one hot custom utility in R4ML
# What is One Hot encoding
# Step 1:
# One hot encoding for the input matrix X
```
dml <- 'Y = table(seq(1, nrow(X), 1), X)'
```
# Step 2:
# Attach the input and output via call to sysml.execute
```
X <- as.r4ml.matrix(as.r4ml.frame(iris$Species))
sysml.outs <- Sysml.execute(dml = dml, X = X, "Y")
```
# Step 3:
# Collect the output
```
Y = sysml.outs[['Y']]```
REAL LIFE EXAMPLES /DEMO
Airline Data Analysis using R4ML

• Do exploratory analysis and manual feature selection for the real dataset

• Pre-process the inputs and do proper na removal, recoding etc

• PCA examples:

• SVM example: (predict airline delay using the SVM)
  • See the results
  • Talks about wider table support in R4ML and systemML

• Illustrate the cross validation and model tuning
Bridging the gap for the Wide columns datasets

- Many Dataset when done one hot encoding, creates many columns
- Many image processing applications requires many columns
- So one of the distinguishing feature of the systemML and R4ML is the support for the wider columns.
Summary
Future work

• More ML algorithm support.

• We will have many real world use cases, examples, and PDF demos.

• Many more new work items are in the pipeline. Please see the latest changes at https://github.com/SparkTC/r4ml

• Other useful links
  • SparkR: https://spark.apache.org/docs/latest/sparkr.html
Summary

- Distributed ML is a complex problem and Apache SystemML makes it easier and we have created a R frontend which works with underlying SparkR and the whole R ecosystem as well.

- R4ML is an open-source scalable machine learning package for implementing end-to-end ML flows.

- It bridges important gaps in SparkR and uses the best of Apache SystemML and SparkR to make data scientists and analysts get the most out of big data analytics.

- R4ML advantages:
  - More ML algorithms
  - Ability to write custom algorithms
  - Wide table data set (images, etc.)
  - Commonly used pre-processors
Thank You!

Thanks to the R4ML Team!

https://github.com/sparkTC/r4ml