Scalable Machine Learning in R with H2O

Erin LeDell @ledell

DSC July 2016
Introduction

• Statistician & Machine Learning Scientist at H2O.ai in Mountain View, California, USA
• Ph.D. in Biostatistics with Designated Emphasis in Computational Science and Engineering from UC Berkeley (focus on Machine Learning)
• Written a handful of machine learning R packages
Agenda

• Who/What is H2O?
• H2O Platform
  • H2O Distributed Computing
  • H2O Machine Learning
• H2O in R
H2O.ai, the Company

- Team: 60; Founded in 2012
- Mountain View, CA
- Stanford & Purdue Math & Systems Engineers

H2O, the Platform

- Open Source Software (Apache 2.0 Licensed)
- R, Python, Scala, Java and Web Interfaces
- Distributed Algorithms that Scale to Big Data
Dr. Trevor Hastie
- John A. Overdeck Professor of Mathematics, Stanford University
- PhD in Statistics, Stanford University
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Co-author with John Chambers, *Statistical Models in S*
- Co-author, *Generalized Additive Models*

Dr. Robert Tibshirani
- Professor of Statistics and Health Research and Policy, Stanford University
- PhD in Statistics, Stanford University
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Author, *Regression Shrinkage and Selection via the Lasso*
- Co-author, *An Introduction to the Bootstrap*

Dr. Steven Boyd
- Professor of Electrical Engineering and Computer Science, Stanford University
- PhD in Electrical Engineering and Computer Science, UC Berkeley
- Co-author, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*
- Co-author, *Linear Matrix Inequalities in System and Control Theory*
- Co-author, *Convex Optimization*
H2O Platform
H2O Platform Overview

- Distributed implementations of cutting edge ML algorithms.
- Core algorithms written in high performance Java.
- APIs available in R, Python, Scala, REST/JSON.
- Interactive Web GUI.
H2O Platform Overview

• Write code in high-level language like R (or use the web GUI) and output production-ready models in Java.
• To scale, just add nodes to your H2O cluster.
• Works with Hadoop, Spark and your laptop.
**H2O Cluster**

- Multi-node cluster with shared memory model.
- All computations in memory.
- Each node sees only some rows of the data.
- No limit on cluster size.

- Distributed data frames (collection of distributed arrays).
- Columns are distributed across the cluster.
- Single row is on a single machine.
- Syntax is the same as R’s data.frame or Python’s pandas.DataFrame.

**H2O Frame**
H2O Communication

Network Communication
- H2O requires network communication to JVMs in unrelated process or machine memory spaces.
- Performance is network dependent.

Reliable RPC
- H2O implements a reliable RPC which retries failed communications at the RPC level.
- We can pull cables from a running cluster, and plug them back in, and the cluster will recover.

Optimizations
- Message data is compressed in a variety of ways (because CPU is cheaper than network).
- Short messages are sent via 1 or 2 UDP packets; larger message use TCP for congestion control.
Data Processing in H2O

Map Reduce

- Map/Reduce is a nice way to write blatantly parallel code; we support a particularly fast and efficient flavor.
- Distributed fork/join and parallel map: within each node, classic fork/join.

Group By

- We have a GroupBy operator running at scale.
- GroupBy can handle millions of groups on billions of rows, and runs Map/Reduce tasks on the group members.

Ease of Use

- H2O has overloaded all the basic data frame manipulation functions in R and Python.
- Tasks such as imputation and one-hot encoding of categoricals is performed inside the algorithms.
H2O on Spark

Sparkling Water

• Sparkling Water is transparent integration of H2O into the Spark ecosystem.
• H2O runs inside the Spark Executor JVM.

Features

• Provides access to high performance, distributed machine learning algorithms to Spark workflows.
• Alternative to the default MLlib library in Spark.
SparkR Implementation Details

- Central controller:
  - Explicitly “broadcast” auxiliary objects to worker nodes

- Distributed workers:
  - Scala code spans Rscript processes
  - Scala communicates with worker processes via stdin/stout using custom protocol
  - Serializes data via R serialization, simple binary serialization of integers, strings, raw byes

- Hides distributed operations
  - Same function names for local and distributed computation
  - Allows same code for simple case, distributed case
H2O vs SparkR

• Although SparkML / MLlib (in Scala) supports a good number of algorithms, SparkR still only supports GLMs.

• Major differences between H2O and Spark:
  • In SparkR, R each worker has to be able to access local R interpreter.
  • In H2O, there is only a (potentially local) instance of R driving the distributed computation in Java.
H2O Machine Learning
Current Algorithm Overview

Statistical Analysis
• Linear Models (GLM)
• Naïve Bayes

Ensembles
• Random Forest
• Distributed Trees
• Gradient Boosting Machine
• R Package - Stacking / Super Learner

Deep Neural Networks
• Multi-layer Feed-Forward Neural Network
• Auto-encoder
• Anomaly Detection
• Deep Features

Clustering
• K-Means

Dimension Reduction
• Principal Component Analysis
• Generalized Low Rank Models

Solvers & Optimization
• Generalized ADMM Solver
• L-BFGS (Quasi Newton Method)
• Ordinary Least-Square Solver
• Stochastic Gradient Descent

Data Munging
• Scalable Data Frames
• Sort, Slice, Log Transform
H2O in R
h2o R Package

Installation

- Java 7 or later; R 3.1 and above; Linux, Mac, Windows
- The easiest way to install the h2o R package is CRAN.
- Latest version: [http://www.h2o.ai/download/h2o/r](http://www.h2o.ai/download/h2o/r)

Design

All computations are performed in highly optimized Java code in the H2O cluster, initiated by REST calls from R.
> library(h2o)
> localH2O <- h2o.init(nthreads = -1, max_mem_size = "8G")

H2O is not running yet, starting it now...

Note: In case of errors look at the following log files:
/var/folders/2j/jg4s153d5q53tc2_nzm9fz5h0000gn/T//RtmpAXY9gj/h2o_me_started_from_r.out
/var/folders/2j/jg4s153d5q53tc2_nzm9fz5h0000gn/T//RtmpAXY9gj/h2o_me_started_from_r.err

java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

Successfully connected to http://127.0.0.1:54321/

R is connected to the H2O cluster:
  H2O cluster uptime: 1 seconds 96 milliseconds
  H2O cluster version: 3.3.0.99999
  H2O cluster name: H2O_started_from_R_me_kfo618
  H2O cluster total nodes: 1
  H2O cluster total memory: 7.11 GB
  H2O cluster total cores: 8
  H2O cluster allowed cores: 8
  H2O cluster healthy: TRUE
library(h2o)  # First install from CRAN
localH2O <- h2o.init()  # Initialize the H2O cluster

# Data directly into H2O cluster (avoids R)
train <- h2o.importFile(path = "train.csv")

# Data into H2O from R data.frame
train <- as.h2o(my_df)
Train a Model & Predict

Example

```r
y <- "Class"
x <- setdiff(names(train), y)

fit <- h2o.gbm(x = x, y = y, training_frame = train)
pred <- h2o.predict(fit, test)
```
Example

```r
hidden_opt <- list(c(200,200), c(100,300,100), c(500,500))
l1_opt <- c(1e-5,1e-7)
hyper_params <- list(hidden = hidden_opt, l1 = l1_opt)

model_grid <- h2o.grid("deplearning",
    hyper_params = hyper_params,
    x = x, y = y,
    training_frame = train,
    validation_frame = test)
```
library(h2oEnsemble)  #Install from GitHub

learner <- c("h2o.randomForest.1",
             "h2o.deeplearning.1",
             "h2o.deeplearning.2")

metalearnern <- "h2o.glm.wrapper"

family <- "binomial"
plot(fit) plots scoring history over time.
H2O R Code


https://github.com/h2oai/h2o-3/blob/26017bd1f5e0f025f6735172a195df4e794f311a/h2o-r/h2o-package/R/models.R#L103
H2O Resources

• H2O Online Training:  http://learn.h2o.ai
• H2O Tutorials:  https://github.com/h2oai/h2o-tutorials
• H2O Slidedecks:  http://www.slideshare.net/0xdata
• H2O Video Presentations:  https://www.youtube.com/user/0xdata
• H2O Community Events & Meetups:  http://h2o.ai/events
The “Intro to H2O” tutorial introduces five popular supervised machine learning algorithms in the context of a binary classification problem.

The training module demonstrates how to train models and evaluating model performance on a test set.

- Generalized Linear Model (GLM)
- Random Forest (RF)
- Gradient Boosting Machine (GBM)
- Deep Learning (DL)
- Naive Bayes (NB)
The second training module demonstrates how to find the best set of model parameters for each model using Grid Search.
http://www.h2o.ai/docs
@ledell on Github, Twitter
erin@h2o.ai

http://www.stat.berkeley.edu/~ledell