Intro to Machine Learning in H2O

Berkeley D-Lab
April 2016

Erin LeDell Ph.D.

H2O.ai

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)
- Worked as a data scientist at several startups
- Written several machine learning software packages

- What/who is H2O?
- H2O Platform
- H2O in R & Python
- H2O Ensemble
- R & Python Demos

H2O.ai

H2O Company

- Team: 55. Founded in 2012, Mountain View, CA
- Stanford Math & Systems Engineers

H2O Software

- Open Source Software
- Ease of Use via Web Interface
- R, Python, Scala, Spark & Hadoop Interfaces
- Distributed Algorithms Scale to Big Data

Scientific Advisory Council

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*
- 108,404 citations (via Google Scholar)

Dr. Rob Tibshirani

- Professor of Statistics and Health Research and Policy, Stanford University
- PhD in Statistics, Stanford University
- COPPS Presidents' Award recipient
- 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. Stephen Boyd

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

H2O Platform

Part 1 of 4
Scalable Machine Learning with H2O

H2O Platform

H2O is an open source, distributed machine learning platform written in Java designed for “big data.”

APIs are available for:
R, Python, Scala & REST/JSON

H2O Platform Overview

Speed Matters!

- Time is valuable
- In-memory is faster
- Distributed is faster
- High speed AND accuracy

No Sampling

- Scale to big data
- Access data links
- Use all data without sampling

Interactive UI

- Web-based modeling with H2O Flow
- Model comparison

Cutting-Edge Algorithms

- Suite of cutting-edge machine learning algorithms
- Deep Learning & Ensembles
- NanoFast Scoring Engine

Current Algorithm Overview

Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

Ensembles

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

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 Distributed Computing

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.

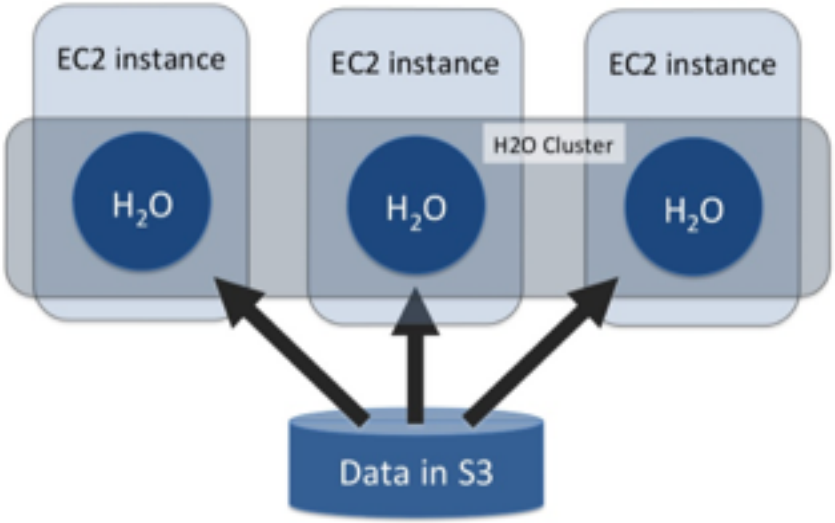
Distributed Key Value Store

- Objects in the H2O cluster such as data frames, models and results are all referenced by key.
- Any node in the cluster can access any object in the cluster by key.

H2O Frame

- Distributed data frames (collection of vectors).
- Columns are distributed (across nodes) arrays.
- Each node must be able to see the entire dataset (achieved using HDFS, S3, or multiple copies of the data if it is a CSV file).

H2O on Amazon EC2



H2O can easily be deployed on an Amazon EC2 cluster.
The GitHub repository contains example scripts that
help to automate the cluster deployment.

Distributed H2O Frame

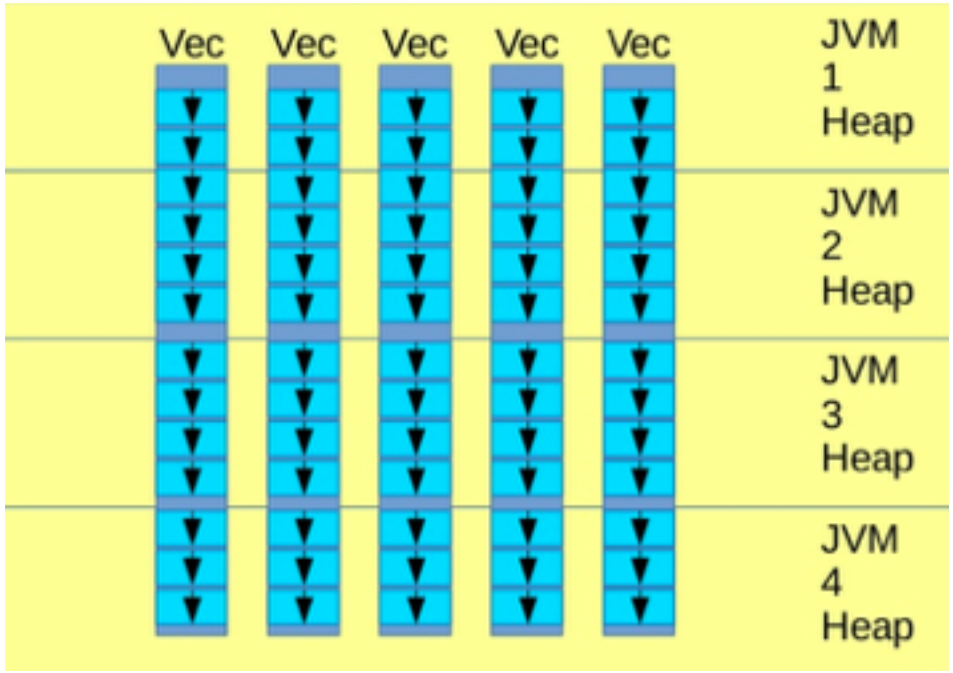


Diagram of distributed arrays. An “H2O Frame” is a collection of distributed arrays.

H2O

GITTER JOIN CHAT →

H2O scales statistics, machine learning, and math over Big Data.

H2O uses familiar interfaces like R, Python, Scala, the Flow notebook graphical interface, Excel, & JSON so that Big Data enthusiasts & experts can explore, munge, model, and score datasets using a range of algorithms including advanced ones like Deep Learning. H2O is extensible so that developers can add data transformations and model algorithms of their choice and access them through all of those clients.

Data collection is easy. Decision making is hard. H2O makes it fast and easy to derive insights from your data through faster and better predictive modeling. H2O allows online scoring and modeling in a single platform.

https://github.com/h2oai/h2o-3

H2O in R & Python

Part 2 of 4
Scalable Machine Learning with H2O

h2o R package

Requirements

- The only requirement to run the “h2o” R package is R >=3.1.0 and Java 7 or later.
- Linux, OS X and Windows.

Installation

- The easiest way to install the “h2o” R package is to install directly from CRAN.
- Latest version: `http://h2o.ai/download`

Design

- No computation is ever performed in R.
- All computations are performed (in highly optimized Java code) in the H2O cluster and initiated by REST calls from R.

Download

Use H_2O directly from R

Copy and paste these commands into R one line at a time:

```
# The following two commands remove any previously installed H2O packages for R.
if ("package:h2o" %in% search()) { detach("package:h2o", unload=TRUE) }
if ("h2o" %in% rownames(installed.packages())) { remove.packages("h2o") }

# Next, we download packages that H2O depends on.
pkgs <- c("methods","statmod","stats","graphics","RCurl","jsonlite","tools","utils")
for (pkg in pkgs) {
    if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}

# Now we download, install and initialize the H2O package for R.
install.packages("h2o", type="source", repos=(c("http://h2o-release.s3.amazonaws.com/h2o/rel-tibshirani/8/R")))
library(h2o)
localH2O = h2o.init()

# Finally, let's run a demo to see H2O at work.
demo(h2o.kmeans)
```

http://h2o.ai/download/h2o/r

h2o Python module

Requirements

- Java 7 or later.
- Python 2 or 3.
- A few Python module dependencies.
- Linux, OS X or Windows.

Installation

- The easiest way to install the “h2o” Python module is PyPI (pip install).
- Latest version: `http://h2o.ai/download`

Design

- No computation is ever performed in Python.
- All computations are performed in highly optimized Java code in the H2O cluster and initiated by REST calls from Python.

Download

Use H2O directly from Python

1. Prerequisite: Python 2.7

2. Install dependencies (prepending with `sudo` if needed):

```
[sudo] pip install requests
[sudo] pip install tabulate
```

At the command line, copy and paste these commands one line at a time:

```
# The following command removes the H2O module for Python.
[sudo] pip uninstall h2o
# Next, use pip to install this version of the H2O Python module.
[sudo] pip install http://h2o-release.s3.amazonaws.com/h2o/rel-tibshirani/8/Python/h2o-3.6.0.8-py2.py3-none-any.whl
```

http://h2o.ai/download/h2o/python

Start H2O Cluster 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/jg4sl53d5q53tc2_nzm9fz5h0000gn/T//RtmpAXY9gj/h2o_me_started_from_r.out
    /var/folders/2j/jg4sl53d5q53tc2_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

>
```

H2O in R: Load Data

Example

```
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)
```

R code example: Load data

H2O in R: Train & Test

Example

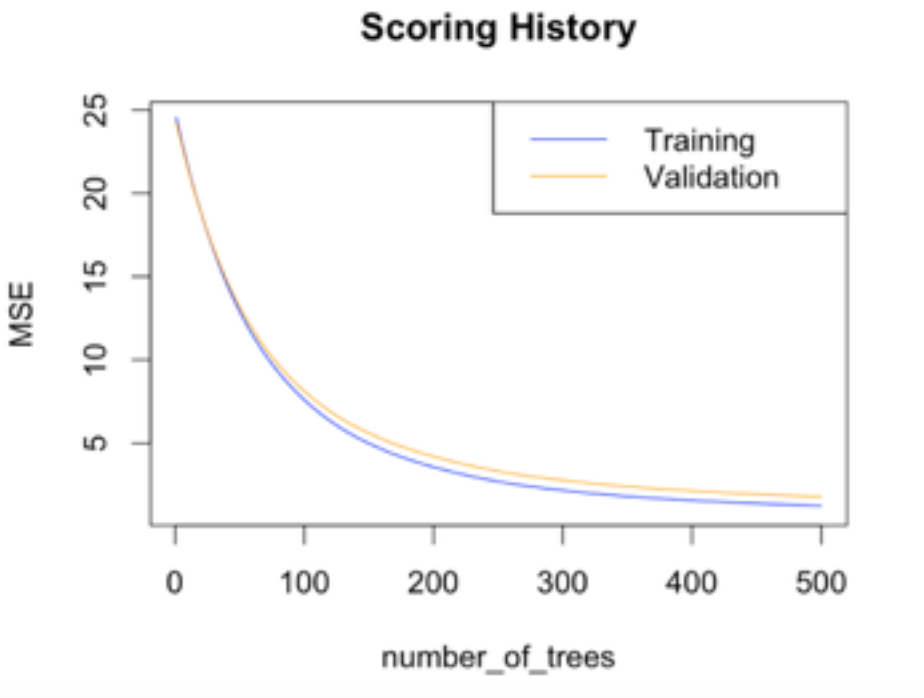
```
y <- "Class"
x <- setdiff(names(train), y)

fit <- h2o.gbm(x = x, y = y, training_frame = train)

pred <- h2o.predict(fit, test)
```

R code example: Train and Test a GBM

H2O in R: Plotting



`plot(fit)` plots scoring history over time.

H2O in R: Grid Search

Example

```
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("deeplearning",
        hyper_params = hyper_params,
        x = x, y = y,
        training_frame = train,
        validation_frame = test)
```

R code example: Execute a DL Grid Search

Start H2O Cluster from Python

```
In [1]: import h2o

        # Start an H2O Cluster on your local machine
        h2o.init()
```

```
No instance found at ip and port: localhost:54321. Trying to start local jar...


JVM stdout: /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T/tmpsfXfv2/h2o_me_started_from_py
thon.out
JVM stderr: /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T/tmpnySTva/h2o_me_started_from_py
thon.err
Using ice_root: /var/folders/2j/jg4sl53d5q53tc2_nzm9fz5h0000gn/T/tmpUU8e_0


Java Version: 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)


Starting H2O JVM and connecting: ..... Connection successful!
```

Start H2O Cluster from Python

In [2]:
```
import h2o

# Start an H2O Cluster on your local machine
h2o.init()
```

H2O cluster uptime:	12 minutes 16 seconds 831 milliseconds
H2O cluster version:	3.6.0.3
H2O cluster name:	H2O_started_from_python
H2O cluster total nodes:	1
H2O cluster total memory:	3.56 GB
H2O cluster total cores:	8
H2O cluster allowed cores:	8
H2O cluster healthy:	True
H2O Connection ip:	127.0.0.1
H2O Connection port:	54321

Train a model (e.g. GBM)

In [23]:
```
# Import H2O GBM:
from h2o.estimators.gbm import H2OGradientBoostingEstimator
```

We first create a `model` object of class, `"H2OGradientBoostingEstimator"`. This does not actually do any training, it just sets the model up for training by specifying model parameters.

In [24]:
```
model = H2OGradientBoostingEstimator(distribution='bernoulli',
                                     ntrees=100,
                                     max_depth=4,
                                     learn_rate=0.1)
```

In [28]:
```
model.train(x=x, y=y, training_frame=train, validation_frame=valid)
```

```
gbm Model Build Progress: [##################################################] 100%
```

Inspect Model Performance

```
In [27]: print(model)
```

```
Model Details
=============
H2OGradientBoostingEstimator :  Gradient Boosting Machine
Model Key:  GBM_model_python_1448559565749_9080

Model Summary:
```

	number_of_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
	100.0	23614.0	4.0	4.0	4.0	10.0	16.0	14.9

```
ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 0.114026790434
R^2: 0.539835211
LogLoss: 0.376005292812
AUC: 0.936370388939
Gini: 0.872740777878

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.43076103173:
```

	0	1	Error	Rate
0	4102.0	814.0	0.1656	(814.0/4916.0)
1	534.0	3538.0	0.1311	(534.0/4072.0)
Total	4636.0	4352.0	0.15	(1348.0/8988.0)

H2O Ensemble

Part 3 of 4
Scalable Machine Learning with H2O

H2O Ensemble Overview

ML Tasks

- Regression
- Binary Classification / Ranking
- Coming soon: Support for multi-class
- Coming soon: Python support

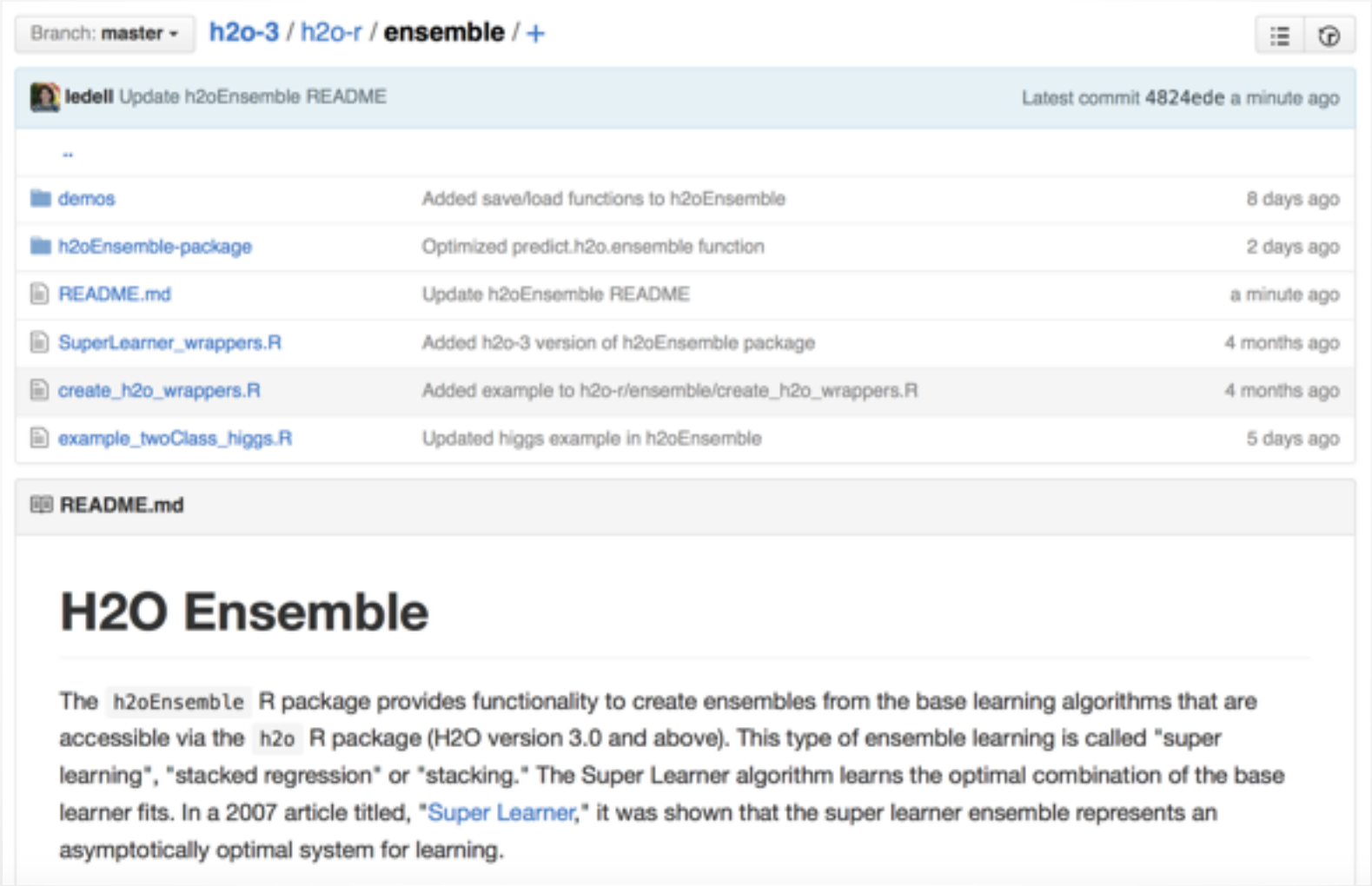
Super Learner

- H2O Ensemble implements the Super Learner algorithm, also called “stacking.”
- The Super Learner algorithm finds the optimal (based on defined loss) combination of a collection of base learning algorithms.

Why ensembles?

- When a single algorithm does not approximate the true prediction function well.
- When model performance is the most important factor (over training speed and interpretability).

h2oEnsemble R package



Branch: **master** ▾ **h2o-3** / h2o-r / **ensemble** / +

ledell Update h2oEnsemble README — Latest commit **4824ede** a minute ago

..		
demos	Added save/load functions to h2oEnsemble	8 days ago
h2oEnsemble-package	Optimized predict.h2o.ensemble function	2 days ago
README.md	Update h2oEnsemble README	a minute ago
SuperLearner_wrappers.R	Added h2o-3 version of h2oEnsemble package	4 months ago
create_h2o_wrappers.R	Added example to h2o-r/ensemble/create_h2o_wrappers.R	4 months ago
example_twoClass_higgs.R	Updated higgs example in h2oEnsemble	5 days ago

README.md

H2O Ensemble

The `h2oEnsemble` R package provides functionality to create ensembles from the base learning algorithms that are accessible via the `h2o` R package (H2O version 3.0 and above). This type of ensemble learning is called "super learning", "stacked regression" or "stacking." The Super Learner algorithm learns the optimal combination of the base learner fits. In a 2007 article titled, "Super Learner," it was shown that the super learner ensemble represents an asymptotically optimal system for learning.

H2O Ensemble R Interface

Example

```
library(h2oEnsemble)  #Install from GitHub

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

metalearner <- "h2o.glm.wrapper"

family <- "binomial"
```

R code example: Set up an H2O Ensemble

H2O Ensemble R Interface

Example

```
fit <- h2o.ensemble(x = x, y = y, training_frame = train,
                    family = family,
                    learner = learner,
                    metalearner = metalearner)

pred <- predict(fit, test)
```

R code example: Train and Test an Ensemble

EEG Demo

Part 4 of 4
Scalable Machine Learning with H2O

EEG for Eye Detection

Problem

- Goal is to accurately predict the eye state using minimal, surface level EEG data.
- Binary outcome: Open vs Closed

Data

- Data from Emotiv Neuralheadset.
- Predictor variables describe signals from 14 EEG channels placed on the surface of the head.

Source: http://archive.ics.uci.edu/ml/datasets/EEG+Eye+State

EEG Data in H2O Flow

eeg_eyestate_splits.hex

Actions: View Data | Split... | Build Model... | Predict | Download | Export

Rows	Columns	Compressed Size
14980	16	837KB

▾ COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
AF3	real	0	0	0	0	1030.7700	309231.0	4321.9178	2492.0722	·	·
F7	real	0	0	0	0	2830.7700	7804.6200	4009.7677	45.9417	·	·
F3	real	0	0	0	0	1040.0	6880.5100	4264.0224	44.4281	·	·
FC5	real	0	0	0	0	2453.3300	642564.0	4164.9463	5216.4046	·	·
T7	real	0	0	0	0	2089.7400	6474.3600	4341.7411	34.7388	·	·
P7	real	0	0	0	0	2768.2100	362564.0	4644.0224	2924.7895	·	·
O1	real	0	0	0	0	2086.1500	567179.0	4110.4002	4600.9265	·	·
O2	real	0	0	0	0	4567.1800	7264.1000	4616.0569	29.2926	·	·
P8	real	0	0	0	0	1357.9500	265641.0	4218.8266	2136.4085	·	·
T8	real	0	0	0	0	1816.4100	6674.3600	4231.3162	38.0509	·	·
FC6	real	0	0	0	0	3273.3300	6823.0800	4202.4569	37.7860	·	·
F4	real	0	0	0	0	2257.9500	7002.5600	4279.2328	41.5443	·	·
F8	real	0	0	0	0	86.6667	152308.0	4615.2053	1208.3700	·	·
AF4	real	0	0	0	0	1366.1500	715897.0	4416.4358	5891.2850	·	·
eyeDetection	int	0	8257	0	0	0	1.0	0.4488	0.4974	·	Convert to enum
split	enum	0	2996	0	0	0	2.0	·	·	3	Convert to numeric

← Previous 20 Columns | → Next 20 Columns

H2O Jupyter Notebooks

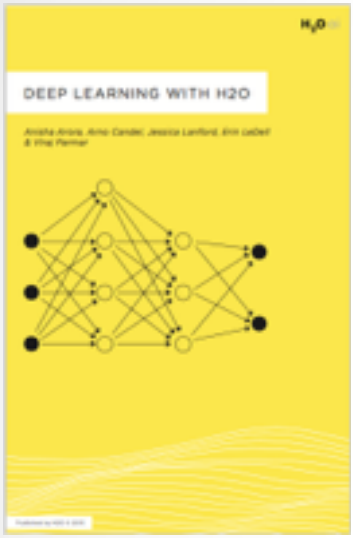
R Notebook: http://tinyurl.com/h2o-eeg-r

Py Notebook: http://tinyurl.com/h2o-eeg-py

Where to learn more?

- H2O Online Training (free): http://learn.h2o.ai
- H2O Slidedecks: http://www.slideshare.net/0xdata
- H2O Video Presentations: https://www.youtube.com/user/0xdata
- H2O Community Events & Meetups: http://h2o.ai/events
- Machine Learning & Data Science courses: http://coursebuffet.com

H2O Booklets



http://tinyurl.com/h2o-github-booklets

Thank you!

@ledell on Twitter, GitHub
erin@h2o.ai

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