Scalable Ensemble Machine Learning

Harvard Health Policy Data Science Lab

Boston, MA April 2017

Erin LeDell Ph.D.
Machine Learning Scientist
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

H2O.ai

Agenda

- Who/What is H2O?
- Ensemble Learning Overview
- Stacking / Super Learner
- Why Stacking?
- Grid Search & Stacking
- Stacking with Third-party Algos
- AutoML and Stacking

H2O.ai

H2O.ai, the Company

- Founded in 2012
- Stanford & Purdue Math & Systems Engineers
- Headquarters: Mountain View, California, USA

H2O, the Platform

- Open Source Software (Apache 2.0 Licensed)
- R, Python, Scala, Java and Web Interfaces
- Distributed Algorithms that 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*

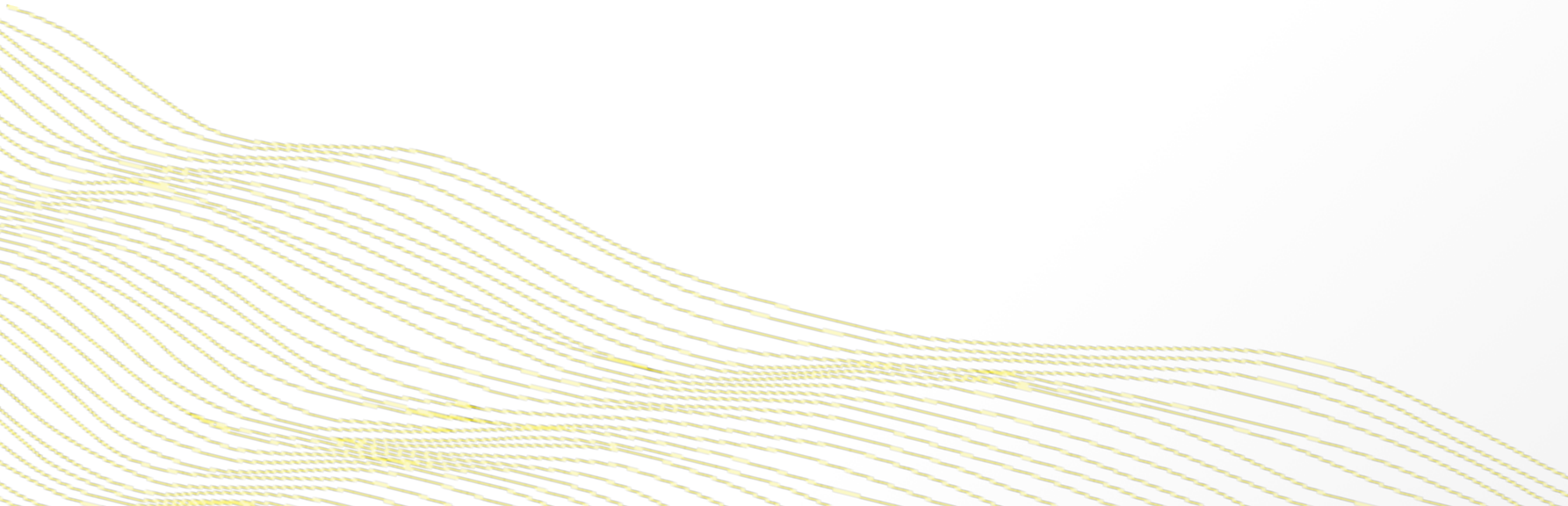
Dr. Robert Tibshirani

- 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*

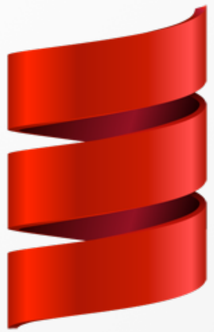
H2O Platform



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 called H2O Flow.
- Easily deploy models to production with H2O Steam.



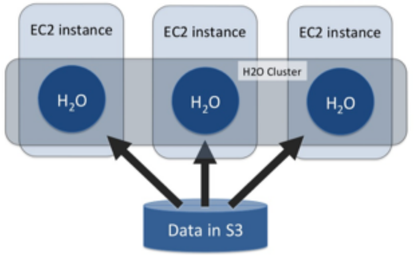




H2O Platform Overview

- 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 Distributed Computing

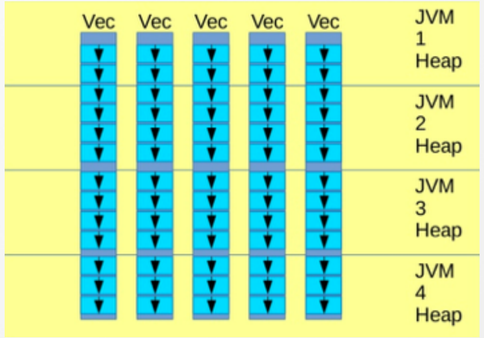
H2O Cluster



- 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.

H2O Frame

- Distributed data frames (collection of vectors).
- Columns are distributed (across nodes) arrays.
- Works just like R's data.frame or Python Pandas DataFrame



Current Algorithm Overview

Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- 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 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

Design

All computations are performed in highly optimized Java code in the H2O cluster, initiated by REST calls from R.

H2O Startup & 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)
```

H2O Machine Learning (e.g. GBM)

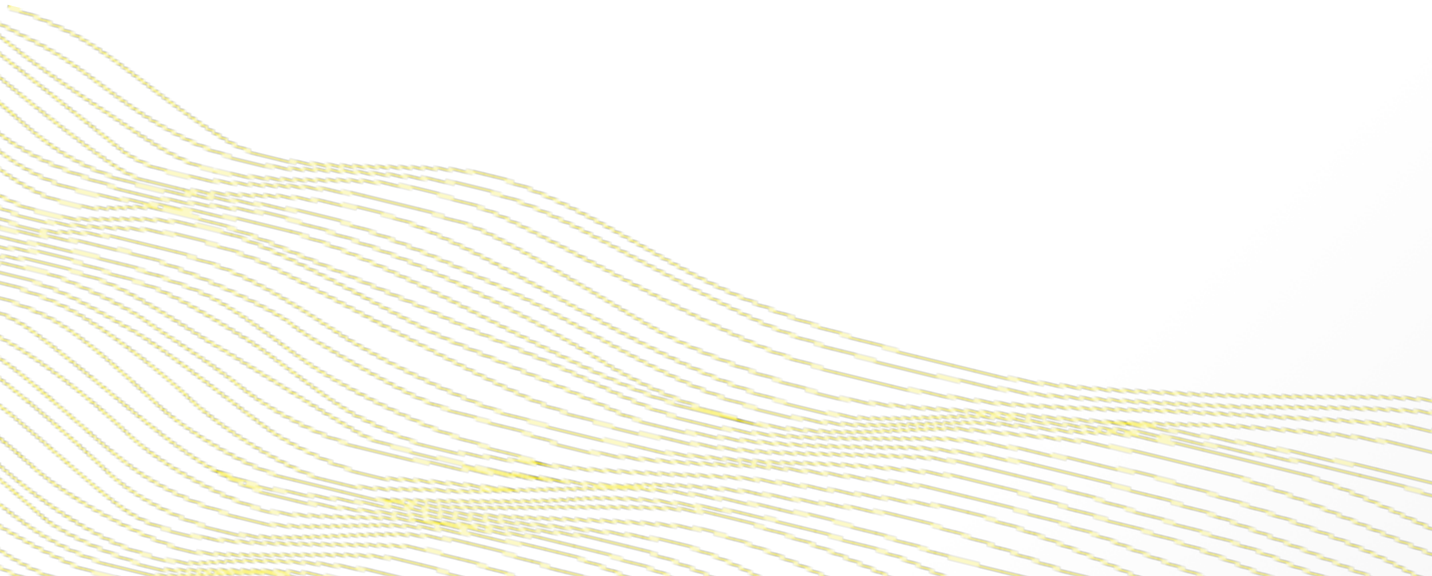
Example

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

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

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

Introduction to Stacking



Ensemble Learning

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained by any of the constituent algorithms.

– Wikipedia

Common Types of Ensemble Methods

Bagging

- Reduces variance and increases accuracy
- Robust against outliers or noisy data
- Often used with Decision Trees (i.e. Random Forest)

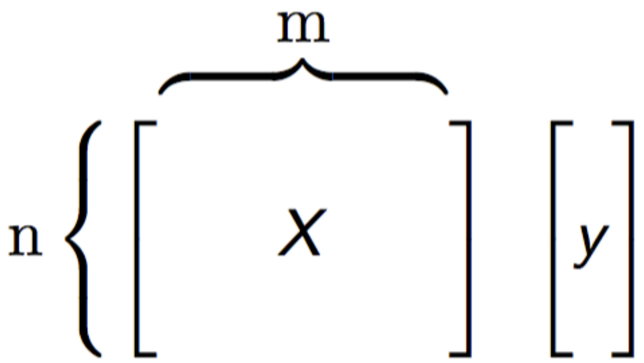
Boosting

- Also reduces variance and increases accuracy
- Not robust against outliers or noisy data
- Flexible – can be used with any loss function

Stacking

- Used to ensemble a diverse group of strong learners
- Involves training a second-level machine learning algorithm called a “metalearner” to learn the optimal combination of the base learners

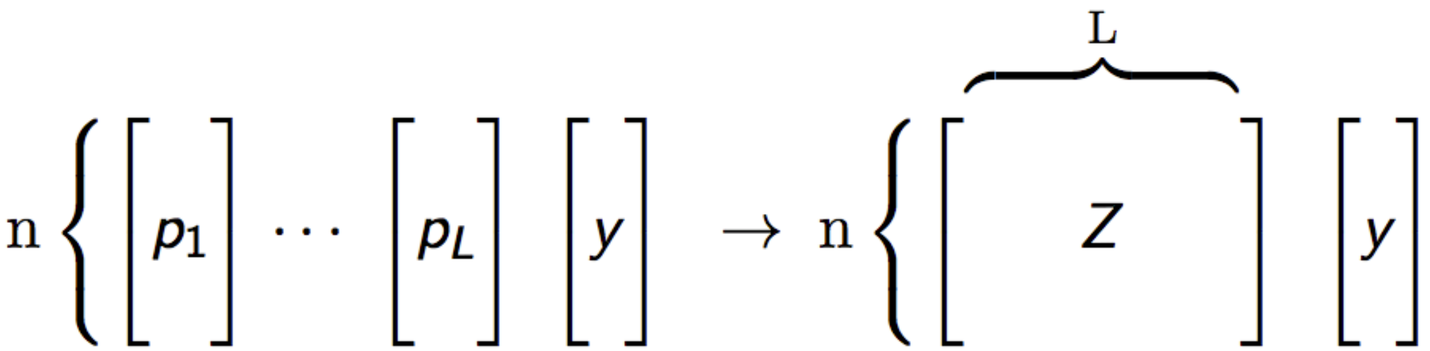
Stacking (aka Super Learner Algorithm)



"Level-zero" data

- Start with design matrix, X, and response, y
- Specify L base learners (with model params)
- Specify a metalearner (just another algorithm)
- Perform k-fold CV on each of the L learners

Stacking (aka Super Learner Algorithm)



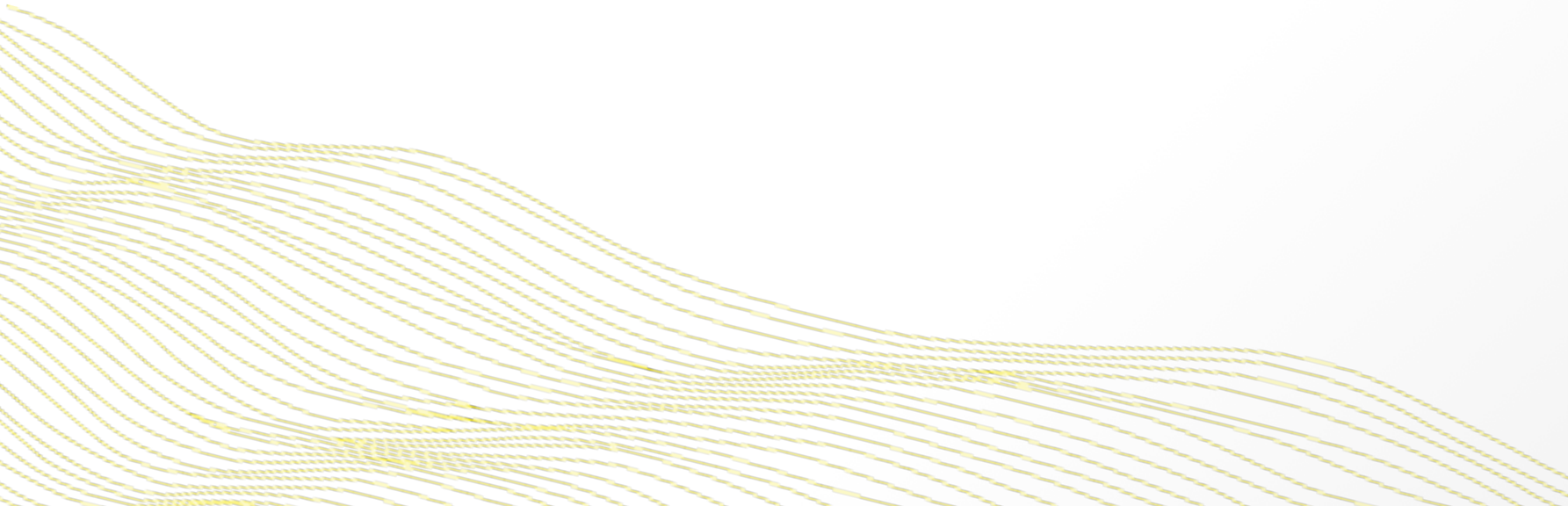
"Level-one" data

- Collect the predicted values from k-fold CV that was performed on each of the L base learners
- Column-bind these prediction vectors together to form a new design matrix, Z
- Train the metalearner using Z, y

Stacking vs. Parameter Tuning/Search

- A common task in machine learning is to perform model selection by specifying a number of models with different parameters.
- An example of this is Grid Search or Random Search.
- The first phase of the Super Learner algorithm is computationally equivalent to performing model selection via cross-validation.
- The latter phase of the Super Learner algorithm (the metalearning step) is just training another single model (no CV).
- With Stacking, your computation does not go to waste!

Software for Super Learning



Super Learner R/Py Software Overview

SuperLearner

- Original Super Learner R implementation (2010).
- Comes with support for many existing machine learning R packages and can be customized to wrap any other.

subsemble

- Implements the Subsemble algorithm for combining models trained on partitions of the data, a variant of Super Learning in R. Like SuperLearner, can be used with any machine learning algorithm in R.

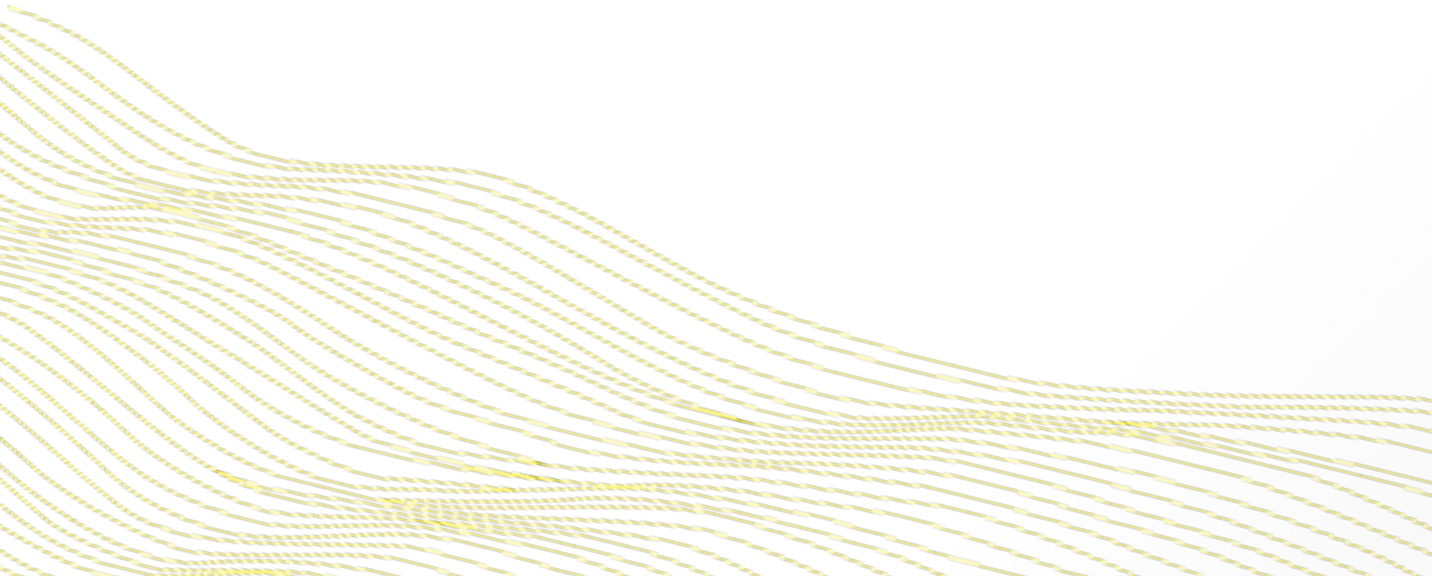
h2oEnsemble

- h2oEnsemble is an R package that implements Super Learner algorithm using H2O distributed algorithms.

h2o

- The h2o R and Python packages now include functions for Super Learning and for automatically creating diverse ensembles.

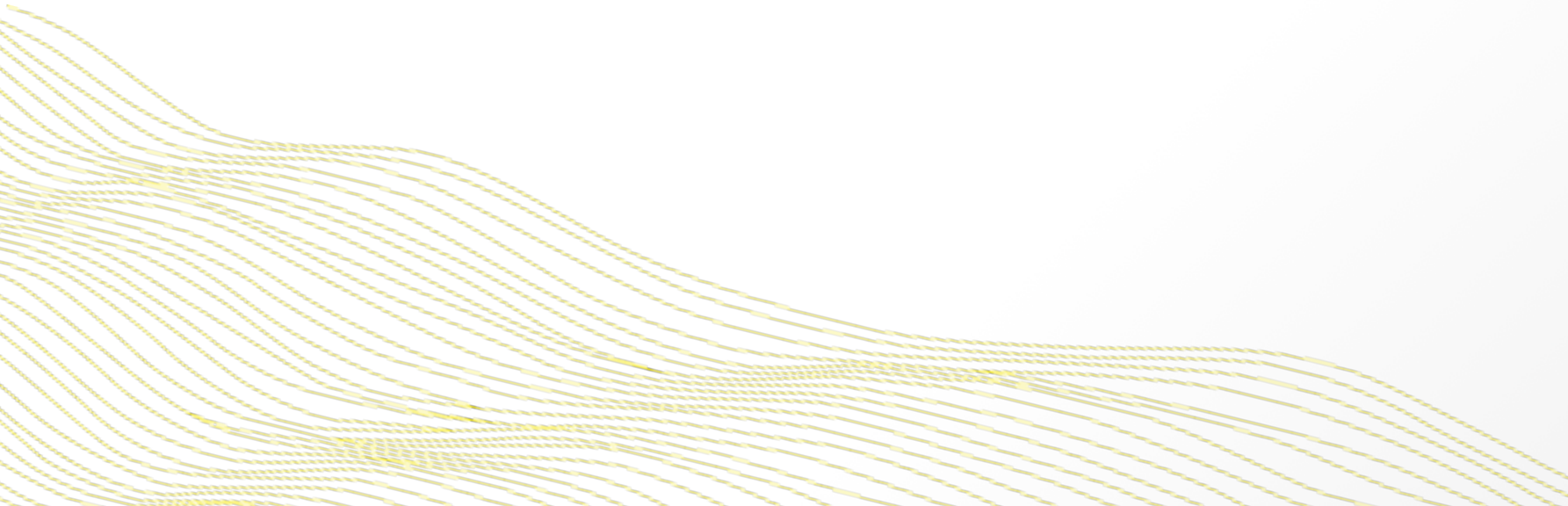
h2oEnsemble R package & Stacked Ensemble in h2o



Evolution of H2O Ensemble

- h2oEnsemble R package in 2015
- Ensemble logic ported to Java in late 2016
- Stacked Ensemble method in h2o in early 2017
  - R & Python APIs supported
  - In progress: Custom metalearners
  - In progress: MOJO for production use

Stacking with Random Grids



H2O Cartesian 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)

grid <- h2o.grid(algorithm = "deeplearning",
                 hyper_params = hyper_params,
                 x = x, y = y,
                 training_frame = train,
                 validation_frame = valid)
```

H2O Random Grid Search

Example

```
search_criteria <- list(strategy = "RandomDiscrete",
                        max_runtime_secs = 600)

grid <- h2o.grid(algorithm = "deeplearning",
                 hyper_params = hyper_params,
                 search_criteria = search_criteria,
                 x = x, y = y,
                 training_frame = train,
                 validation_frame = valid)
```

Stacking with Random Grids (h2o R)

Example

```
# Create a list of all the base models
models <- c(gbm_models, rf_models, dl_models, glm_models)

# Let's stack!
fit <- h2o.stackedEnsemble(x = x, y = y,
                           selection_strategy="choose_all",
                           training_frame = train,
                           base_models = models)
```

H2O Stacking Resources

H2O Stacked Ensembles docs & code demo:
http://tinyurl.com/h2o-stacked-ensembles

h2oEnsemble R package homepage on Github:
http://tinyurl.com/github-h2o-ensemble

Tutorial: Stacked Ensembles

There are two H2O Ensemble tutorials in R: One for the new Stacked Ensemble method in h2o and one for the h2oEnsemble R package which extends the h2o R API.

H2O AutoML

- AutoML stands for "Automatic Machine Learning"
- The idea here is to remove most (or all) of the parameters from the algorithm, as well as automatically generate derived features that will aid in learning.
- Single algorithms are tuned automatically using a carefully constructed random grid search (future: search by Bayesian Optimization).
- Optionally, a Stacked Ensemble can be constructed.

Public code coming soon!

H2O Resources

- H2O Online Training: http://learn.h2o.ai
- H2O Tutorials: https://github.com/h2oai/h2o-tutorials
- H2O Meetup Materials: https://github.com/h2oai/h2o-meetups
- H2O Video Presentations: https://www.youtube.com/user/0xdata
- H2O Community Events & Meetups: https://h2o.ai/events

Thank you!

@ledell on Github, Twitter

erin@h2o.ai

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