

Introduction to H2O



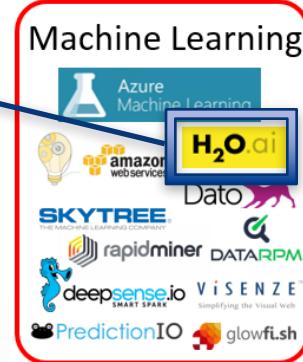
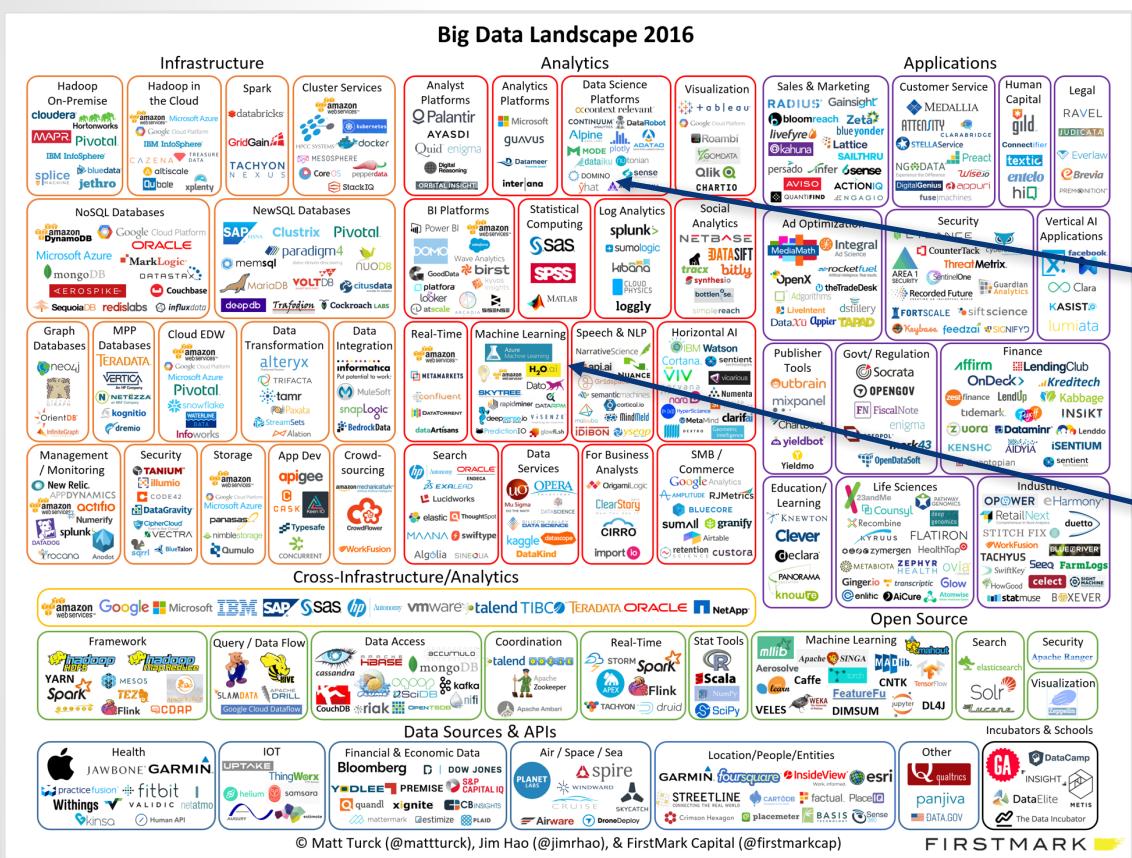
Jo-fai (Joe) Chow
Data Scientist
joe@h2o.ai
@matlabulous

Data Science for IoT Meetup – London
July 2016

About Me: Engineer → Data Scientist

- 2005 - 2015
- Water Engineer
 - Consultant for Utilities
 - SEAMS (Sheffield)
 - EngD Research
 - Water + Machine Learning
 - University of Exeter
 - XP Solutions (Newbury)
- 2015 - Present
- Data Scientist
 - UK Telecom
 - Virgin Media
 - Silicon Valley
 - Domino Data Lab
 - H2O.ai

My Silicon Valley Journey



Agenda

- Who/What is H2O?
- H2O Machine Learning Platform
- H2O in Web, R and Python
- H2O Use Cases
- What's Next?



ABOUT H2O

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*



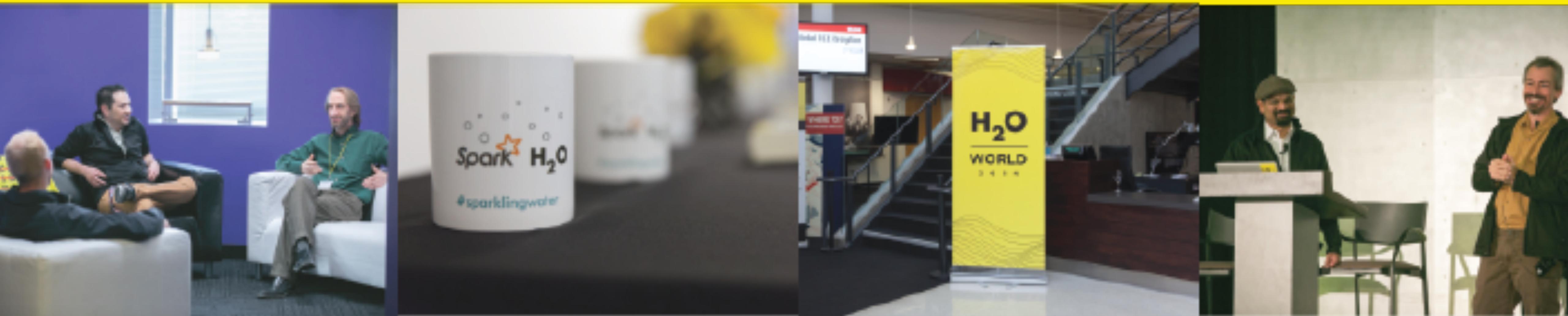
Dr. Robert Tibshirani

- 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.ai, the Company

- Team: 55; 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

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 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.
-

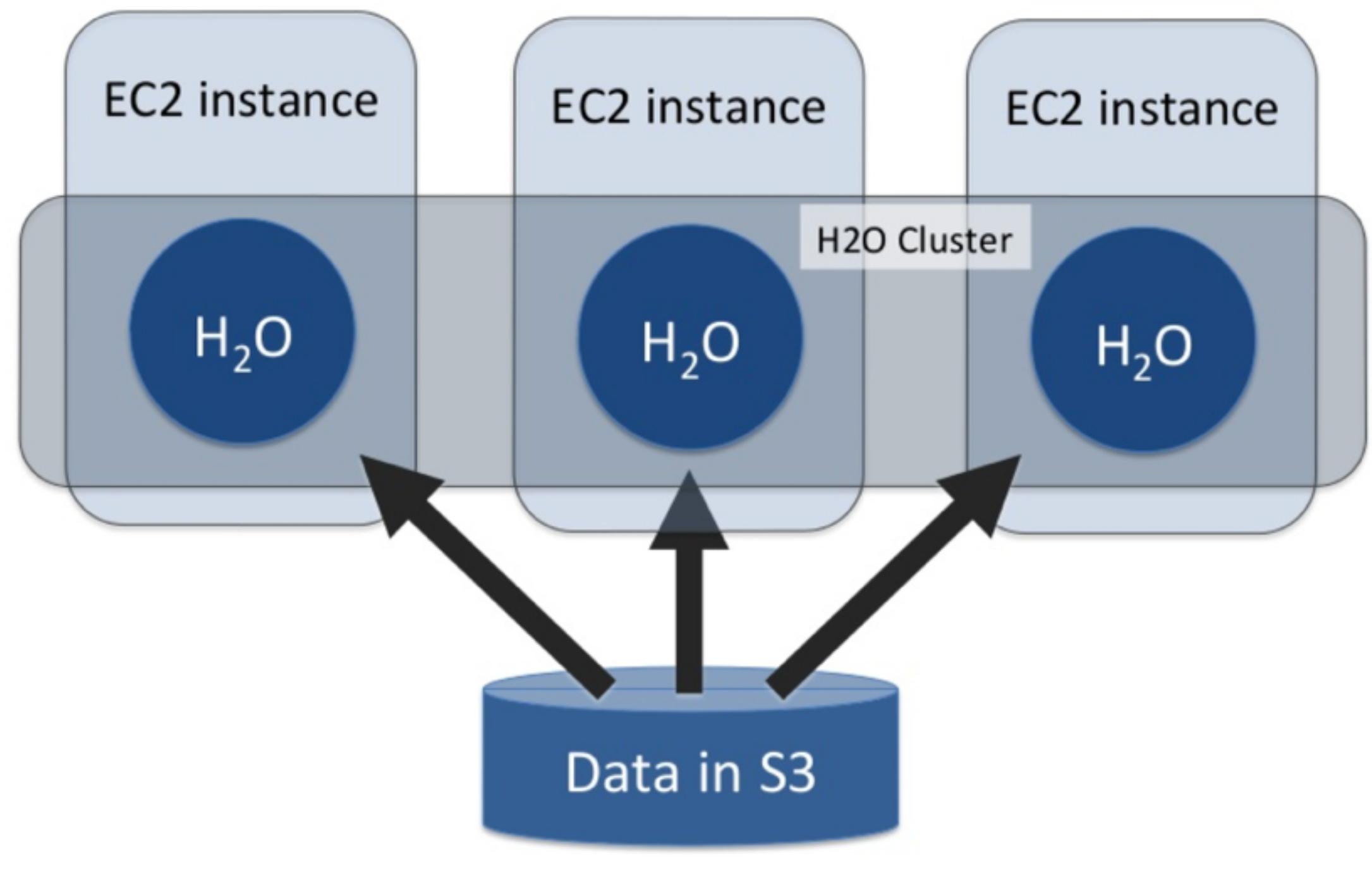
Distributed K/V Store

- 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.
- Works just like R's `data.frame` or Python Pandas `DataFrame`

H2O on Amazon EC2



H2O can easily be deployed on an Amazon EC2 cluster.

The H2O GitHub repository contains example scripts that help to automate the cluster deployment ("ec2" folder).

H2O API FOR R / PYTHON

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 Python Module



Installation

- Java 7 or later; Python 2.7, 3.5; Linux, Mac, Windows
- The easiest way to install the h2o Python module is PyPi.
- Latest version: <http://www.h2o.ai/download/h2o/python>

Design

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

“FLOW” – H2O WEB INTERFACE

Demo – Web Interface

The screenshot shows the H2O Flow web interface running in a browser window. The title bar reads "H2O Flow" and the address bar shows "localhost:54321/flow/index.html". The main menu includes Flow, Cell, Data, Model, Score, Admin, and Help.

The left sidebar is titled "Untitled Flow" and contains a toolbar with various icons for file operations like open, save, copy, paste, and delete, along with navigation arrows and a search icon.

The central area displays a table titled "Assistance" under the heading "cs assist". The table lists routines with their descriptions:

Routine	Description
importFiles	Import file(s) into H2O
getFrames	Get a list of frames in H2O
splitFrame	Split a frame into two or more frames
getModels	Get a list of models in H2O
getGrids	Get a list of grid search results in H2O
getPredictions	Get a list of predictions in H2O
getJobs	Get a list of jobs running in H2O
buildModel	Build a model
importModel	Import a saved model
predict	Make a prediction

The status bar at the bottom indicates "Ready" and "Connections: 0".

The right side of the interface features a "HELP" tab selected in the top navigation bar. The "Help" section contains a heading "Using Flow for the first time?", a "Quickstart Videos" button, and a link "Or, view example Flows to explore and learn H2O." A blue arrow points from this link towards the "GENERAL" section below. The "GENERAL" section lists related topics:

- Flow Web UI ...
- ... Importing Data
- ... Building Models
- ... Making Predictions
- ... Using Flows

Deep Learning Tutorial

The screenshot shows the H2O Flow web application interface. At the top, there's a toolbar with various icons for navigation and file operations. Below the toolbar, the main header reads "H2O FLOW" followed by a menu bar with "Flow", "Cell", "Data", "Model", "Score", "Admin", and "Help". A user name "Jo-fai" is visible in the top right corner.

The main content area displays a "DeepLearning_MNIST" project. On the left, there's a sidebar with a "Deep Learning Tutorial" section containing an MD (Markdown) file. The content of the MD file is:

MD Deep Learning Tutorial

The purpose of this tutorial is to walk new users through Deep Learning using H2O Flow.

Those who have never used H2O before should refer to [Getting Started](#) for additional instructions on how to run H2O Flow.

For tips on improving the performance and results of your Deep Learning model, refer to our [Definitive Performance Tuning Guide for Deep Learning](#).

Using Deep Learning

H2O's Deep Learning functionalities include:

- purely supervised training protocol for regression and classification tasks
- fast and memory-efficient Java implementations based on columnar compression and fine-grain Map/Reduce
- multi-threaded and distributed parallel computation to be run on either a single node or a multi-node cluster
- fully automatic per-weight adaptive learning rate for fast convergence
- optional specification of learning rate, annealing and momentum options
- regularization options include L1, L2, dropout, Hogwild! and model averaging to prevent model overfitting
- grid search for hyperparameter optimization and model selection
- model checkpointing for reduced run times and model tuning
- automatic data pre- and post-processing (one-hot encoding and standardization) for categorical and numerical data
- automatic imputation of missing values
- automatic tuning of communication in computation for best performance

On the right side of the interface, there's a sidebar titled "OUTLINE" which lists several flow files: GBM_Example.flow, DeepLearning_MNIST.flow, GLM_Example.flow, DRF_Example.flow, K-Means_Example.flow, Million_Songs.flow, KDDCup2009_Churn.flow, QuickStartVideos.flow, Airlines_Delay.flow, GBM_Airlines_Classification.flow, and GBM_GridSearch.flow. There's also a "HELP" tab and a "PACK" button.

At the bottom of the interface, there's a status bar with the message "Ready" and "Connections: 0". The H2O logo is also present in the bottom right corner.

Training a Deep Learning Model

The screenshot shows the H2O Flow interface running a "DeepLearning_MNIST" flow. The main area displays the R code for the buildModel command:

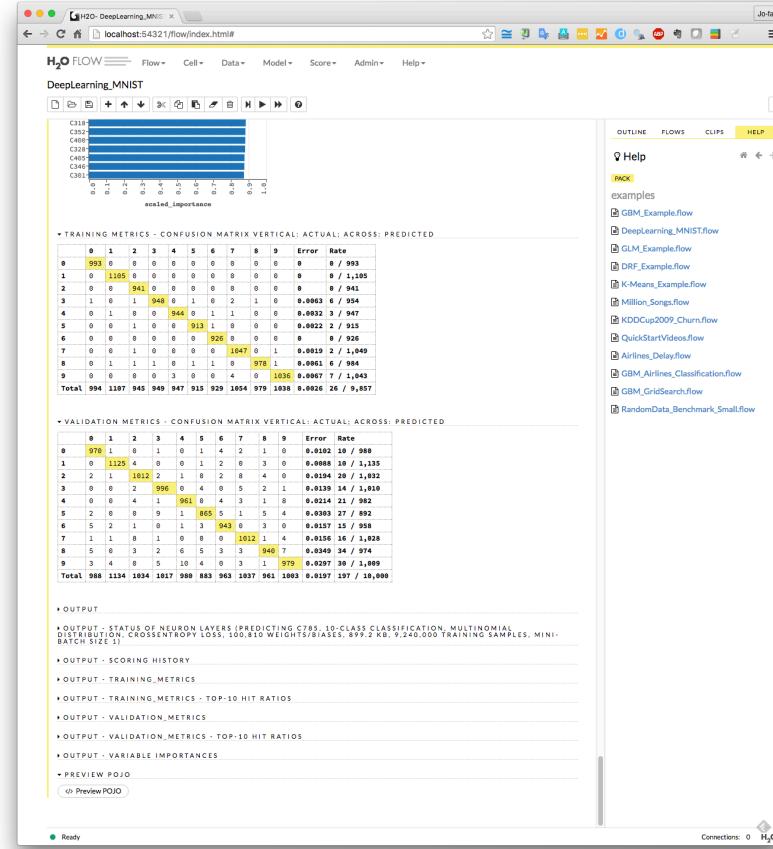
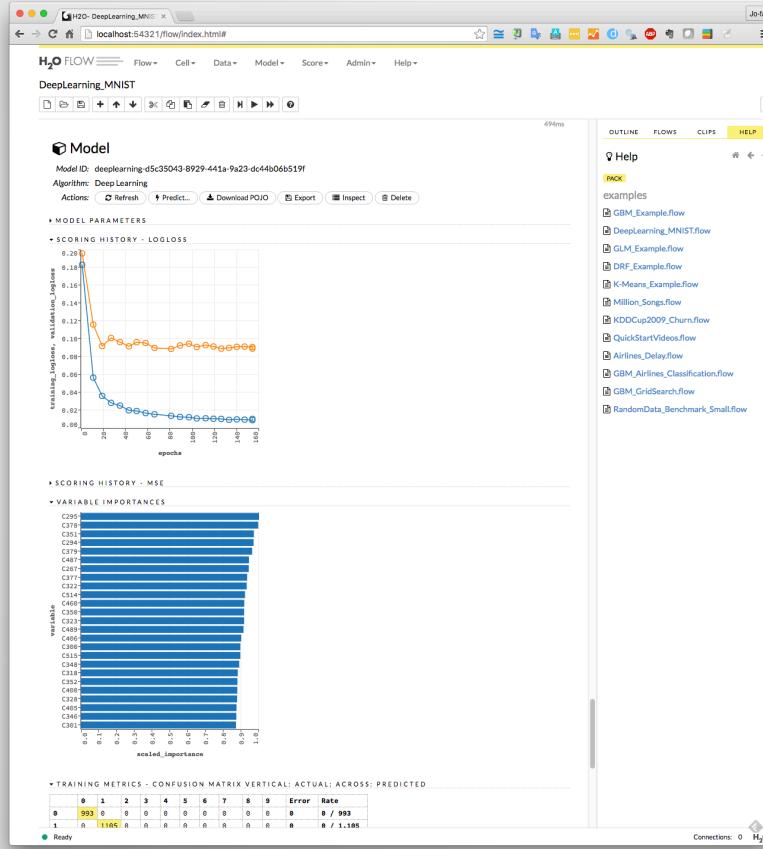
```
buildModel 'deeplearning', {"model_id": "deeplearning-d5c35043-8929-441a-9a23-dc44b06b519f", "training_frame": "train.hex", "validation_frame": "test.hex", "nfolds": 0, "response_column": "C785", "ignored_columns": [], "ignore_const_cols": true, "activation": "RectifierWithDropout", "hidden": [128, 64], "epochs": 500, "variable_importances": true, "balance_classes": false, "max_confusion_matrix_size": 20, "max_hit_ratio_k": 10, "checkpoint": "", "use_all_factor_levels": true, "train_samples_per_iteration": -2, "adaptive_rate": false, "input_dropout_ratio": 0.2, "hidden_dropout_ratios": [0.3, 0.2], "l1": 1e-4, "l2": 1e-4, "loss": "Automatic", "distribution": "AUTO", "score_interval": 5, "score_training_samples": 10000, "score_validation_samples": 0, "score_duty_cycle": 0.1, "stopping_rounds": 3, "stopping_metric": "misclassification", "stopping_tolerance": 1e-2, "autoencoder": false, "overwrite_with_best_model": true, "target_ratio_comm_to_comp": 0.05, "seed": 2517542307834282500, "rate": "0.005", "rate_annealing": 1e-6, "rate_decay": 1, "momentum_start": "0.9", "momentum_ramp": 1e6, "momentum_stable": "0.99", "nesterov_accelerated_gradient": true, "max_w2": "Infinity", "initial_weight_distribution": "UniformAdaptive", "classification_stop": -1, "score_validation_sampling": "Uniform", "diagnostics": true, "fast_mode": true, "force_load_balance": true, "single_node_mode": false, "shuffle_training_data": false, "missing_values_handling": "MeanImputation", "quiet_mode": false, "sparse": true, "col_major": false, "average_activation": 0, "sparsity_beta": 0, "max_categorical_features": 2147483647, "reproducible": false, "export_weights_and_biases": false, "elastic_averaging": false}
```

The status bar at the bottom indicates "Running cell 15 of 17" and "Job Ready". On the right side, there is a sidebar titled "examples" listing various flow files:

- GBM_Example.flow
- DeepLearning_MNIST.flow
- GLM_Example.flow
- DRF_Example.flow
- K-Means_Example.flow
- Million_Songs.flow
- KDDCup2009_Churn.flow
- QuickStartVideos.flow
- Airlines_Delay.flow

At the bottom right, it says "Connections: 0" and has a small H2O logo.

Model Evaluation



Export Plain Old Java Object (POJO)

The screenshot shows the H2O Flow interface with the title "DeepLearning_MNIST". The left panel displays the generated POJO code for a DeepLearningModel named "d5c35043_8929_441a_9a23_dc44b06b519f". The code includes imports for java.util.Map, hex.gm.GemModel, and hex.gm.annotations.ModelPojo; a class definition for "DeepLearning_d5c35043_8929_441a_9a23_dc44b06b519f" extending GemModel; and a static final field "NORMALUL" containing 26 double values ranging from 0.1838 to 1.2263361625631215.

```
/*
 Licensed under the Apache License, Version 2.0
 http://www.apache.org/licenses/LICENSE-2.0.html

 AUTOGENERATED BY H2O at 2016-07-13T13:04:11.112+01:00
 3.8.2.9

 Standalone prediction code with sample test data for DeepLearningModel named deeplearning_d5c35043_8929_441a_9a23_dc44b06b519f

 How to download, compile and execute:
 mkdir tmpdir
 cd tmpdir
 curl -o gm.jar http://127.0.0.1:54321/h2o-genmodel.jar > h2o-genmodel.jar
 curl -o data.zip http://127.0.0.1:54321/H2OData/java/deeplearning-d5c35043_8929_441a_9a23_dc44b06b519f > deeplearning
 java -cp h2o-genmodel.jar -D-Xms2g -D-Xmx2g -DMaxPermSize=128m deeplearning_d5c35043_8929_441a_9a23_dc44b06b519f

 (Note: Try java argument -XX:+PrintCompilation to show runtime JIT compiler behavior.)

 */
import java.util.Map;
import hex.gm.GemModel;
import hex.gm.annotations.ModelPojo;

@ModelPojo(name="deeplearning_d5c35043_8929_441a_9a23_dc44b06b519f", algorithm="deeplearning")
public class deeplearning_d5c35043_8929_441a_9a23_dc44b06b519f extends GemModel {
    public hex.ModelCategory getModelCategory() { return hex.ModelCategory.Multinomial; }
    public boolean isSupervised() { return true; }
    public int nClasses() { return 10; }
    // Thread-local storage for input neuron activation values.
    final double[] NUM = new double[717];
    static final NORMUL implements java.io.Serializable {
        public static final double[] VALUES = new double[717];
        static {
            NORMUL_0.fill(VALUES);
        }
        static final class NORMUL_0 implements java.io.Serializable {
            static void fill(double[] sa) {
                sa[0] = 2.193829171915183;
                sa[1] = 0.7340517801982369;
                sa[2] = 0.179533355869159;
                sa[3] = 27.21655269759863;
                sa[4] = 15.38931892349664;
                sa[5] = 5.137789388071788;
                sa[6] = 0.174204044779924;
                sa[7] = 0.4772012336729927;
                sa[8] = 0.337025194512487;
                sa[9] = 0.3898323599546;
                sa[10] = 0.2053841185467;
                sa[11] = 0.179533355869159;
                sa[12] = 0.179533355869159;
                sa[13] = 0.179533355869159;
                sa[14] = 0.179533355869159;
                sa[15] = 0.179533355869159;
                sa[16] = 0.1674029351277835;
                sa[17] = 0.184726624771580;
                sa[18] = 0.2398285723799588;
                sa[19] = 0.21655269759863;
                sa[20] = 0.3116525827584749;
                sa[21] = 0.3146495951593469;
                sa[22] = 0.4255991494921587;
                sa[23] = 1.2398080599330034;
                sa[24] = 0.174204044779924;
                sa[25] = 0.174204044779924;
                sa[26] = 1.707572683781805;
                sa[27] = 1.2263361625631215;
            }
        }
    }
}
```

The screenshot shows the H2O Flow interface with the title "DeepLearning_MNIST". The right panel displays the "Model" section. It shows the Model ID "deeplearning-d5c35043-8929-441a-9a23-dc44b06b519f" and the Algorithm "Deep Learning". Below these, there are several buttons: Refresh, Predict..., Download POJO (which has a blue arrow pointing to it), Export, Inspect, and Delete.

USE CASES

Use Case – Recommender System



H2O Powers Intelligent Product
Recommendation Engine
at Transamerica
Case Study

Use Case – Recommender System

Inputs

Your Age: 35
Your Gender: Male
Your State: CA
Your ZIP code: 94043
Number of dependents: 2
Household size: 4
Annual Income: 75000

Click to Recommend

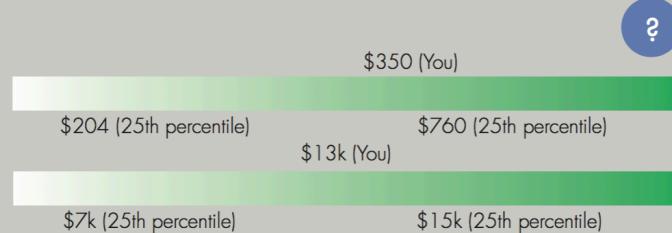
Burial Coverage

Annual Premium:

\$350

Policy Face Value:

\$13,000



Term Coverage

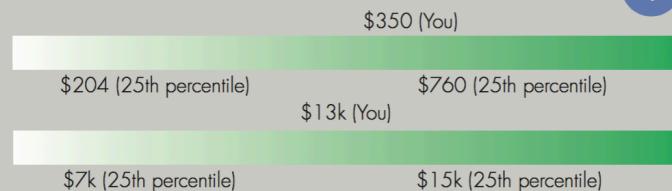
Annual Premium:

\$1,300

Policy Face Value:

\$150,000

15-year Term Life Insurance Policy



More Use Cases



Solving Customer Churn with
Machine Learning

Case Study



Machine Learning to Save Lives
Case Study



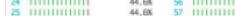
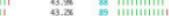
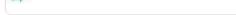
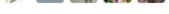
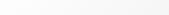
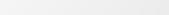
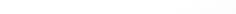
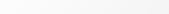
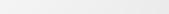
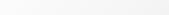
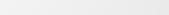
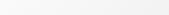
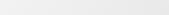
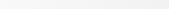
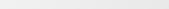
MarketShare Enhances Its Marketing
Analytics Services Using H2O

Case Study

The H2O.ai logo, which features the text 'H2O.ai' in a bold, black, sans-serif font.The H2O.ai logo, which features the text 'H2O.ai' in a bold, black, sans-serif font.The H2O.ai logo, which features the text 'H2O.ai' in a bold, black, sans-serif font.

Use Case – How People Use H2O

 Arno Candel @ArnoCandel	
Great use of H2O #DeepLearning for 3rd place at #Kaggle Homesite challenge! github.com/mpearmain/home... (see R scripts)	
<pre>dl.model <- h2o.deeplearning(# data specifications x = xrange, y = max(xrange)+1, training_frame = x0.hex, autoencoder = FALSE, # network structure: activation and geometry activation = "RectifierWithDropout", hidden = c(size1, size2), epochs = 25, input_dropout_ratio = 0.05, hidden_dropout_ratios = c(0.05, 0.02), # parameters of the optimization process rho = 0.99, epsilon = 1e-08, rate = 0.005, rate_annealing = 1e-06, rate_decay = rate_dec, momentum_start = 0.5, l1 = 0, l2 = 0, loss = c("CrossEntropy"))</pre>	
RETWEETS 18	LIKES 33
 A row of small user profile icons representing the 33 likes.	
1:39 AM - 30 Jun 2016	
 Mountain View, CA	
	 18
 33	•••

	Szilard @DataScienceLA	 Following
Also @h2oai on monster 2TB RAM 128 cores EC2 X1 #bigdata #machinelearning #datascience twitter.com/DataScienceLA/ ...		
 58.1K  42.9K  41.7K  37.8K		
 48.1K  40.0K  39.7K  42.7K		
 48.1K  40.0K  39.7K  42.7K		
 43.9K  44.6K  40.3K  48.0K		
 45.0K  42.9K  40.8K  36.5K		
 45.2K  42.9K  37.0K  34.5K		
 43.0K  40.0K  42.4K  35.3K		
 43.0K  42.9K  45.5K  35.3K		
 42.7K  45.5K  45.5K  36.9K		
 51.0K  44.5K  44.9K  32.1K		
 44.5K  42.9K  44.4K  39.0K		
 44.5K  42.9K  44.3K  40.1K		
 47.1K  41.0K  42.7K  48.0K		
 44.2K  40.0K  42.9K  38.1K		
 44.2K  41.0K  41.4K  34.8K		
 48.7K  41.0K  36.0K  46.0K		
 43.0K  43.9K  45.9K  35.9K		
 42.3K  41.3K  40.3K  29.0K		
 44.5K  42.3K  38.0K  37.4K		
 44.6K  43.3K  36.5K  35.7K		
 45.2K  45.5K  28.7K  35.3K		
 41.0K  45.5K  38.7K  46.0K		
 44.0K  43.0K  37.4K  48.0K		
 44.6K  43.2K  37.8K  41.4K		
 43.7K  40.1K  42.3K  39.7K		
 45.5K  40.0K  42.3K  34.2K		
 44.0K  42.0K  38.0K  37.4K		
 43.9K  41.7K  39.7K  31.4K		
 41.4K  38.5K  43.3K  33.3K		
 43.3K  37.2K  38.5K  39.5K		
46.0K 44.0K 33.0K 44.6K		
Mon 11 Sep	23814/1095298M Total: 25 hr, 307 kthr; 3 running Load average: 70.18 33.24 Uptime: 06:36:00	
RETWEETS 10	LIKES 28	        
5:55 PM - 24 Jun 2016		
 10	 28	       

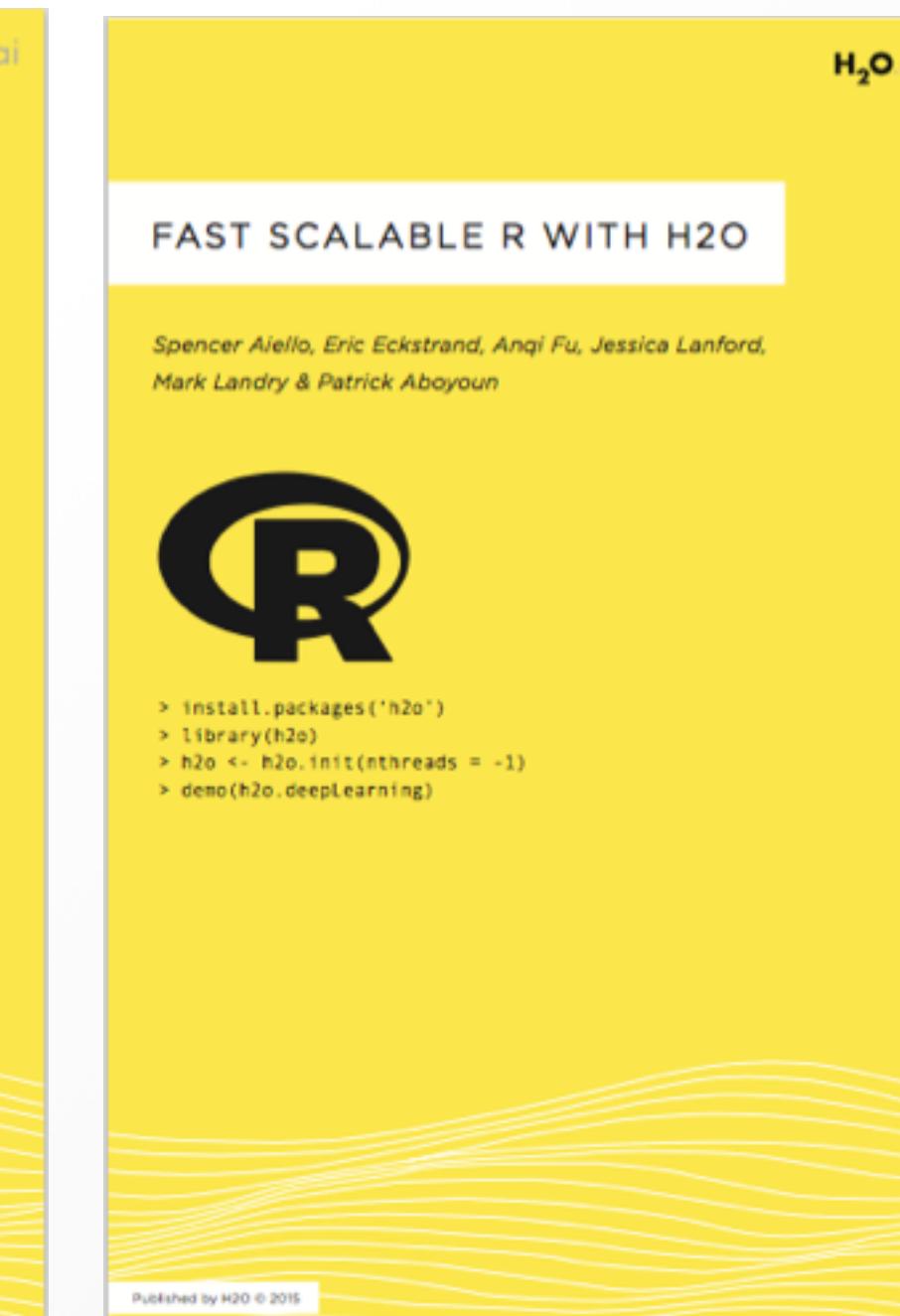
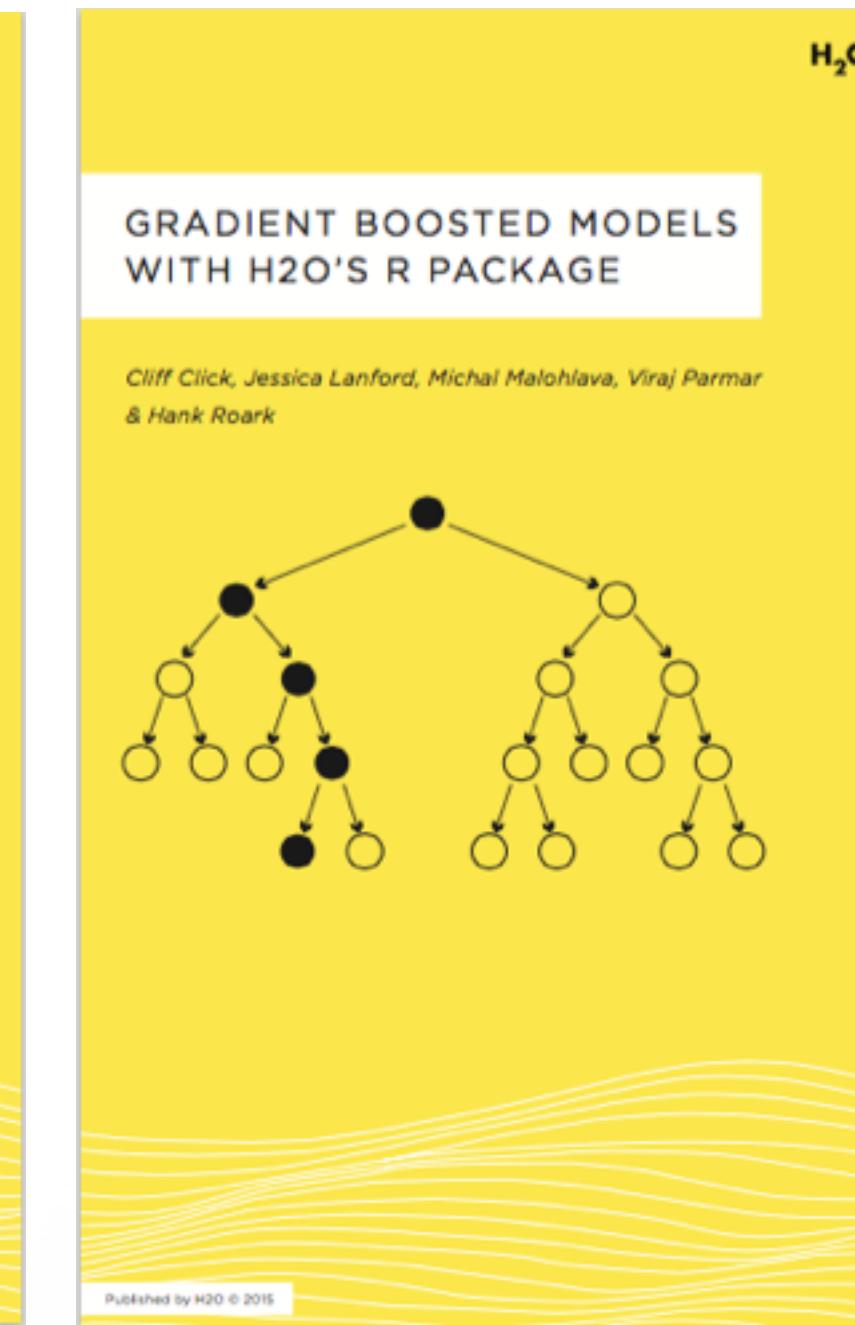
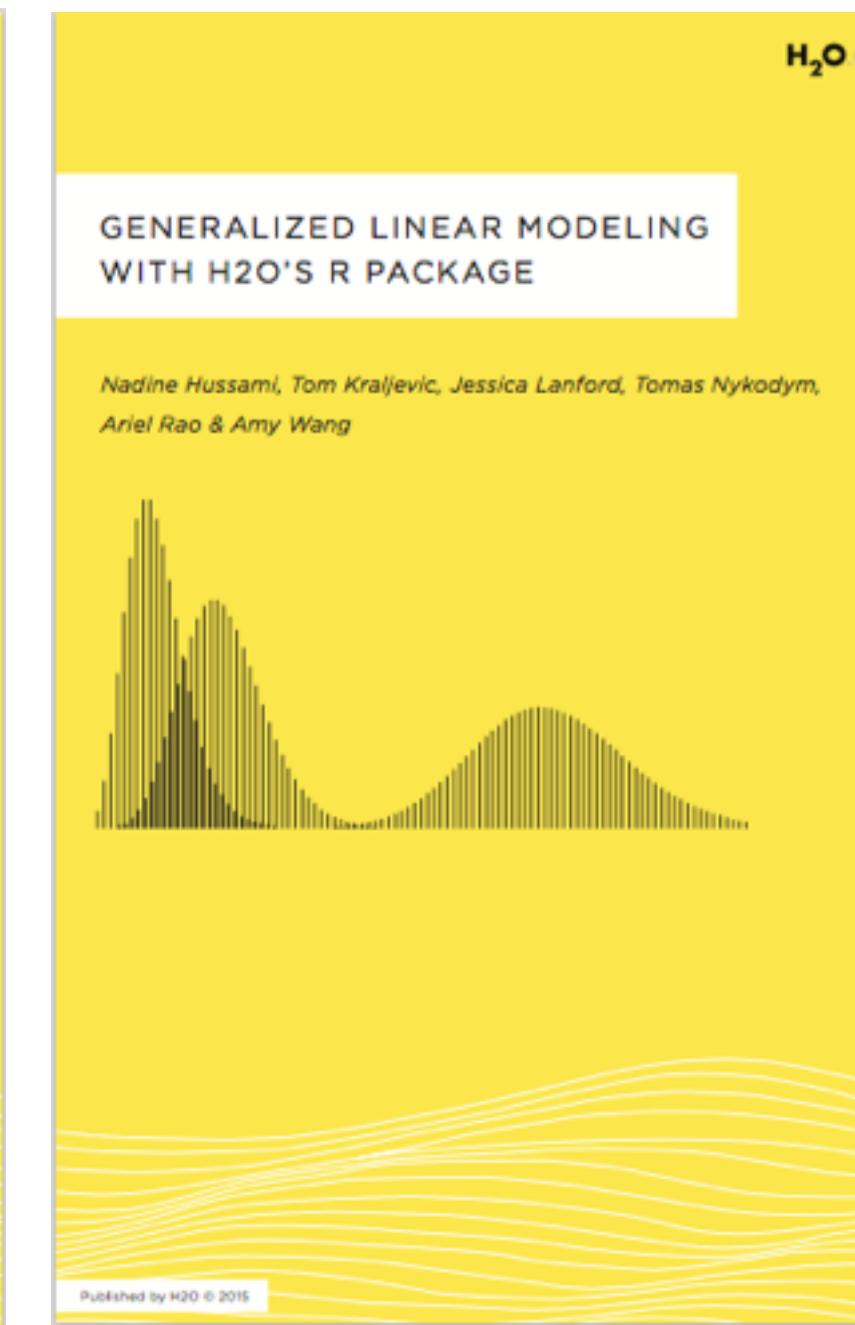
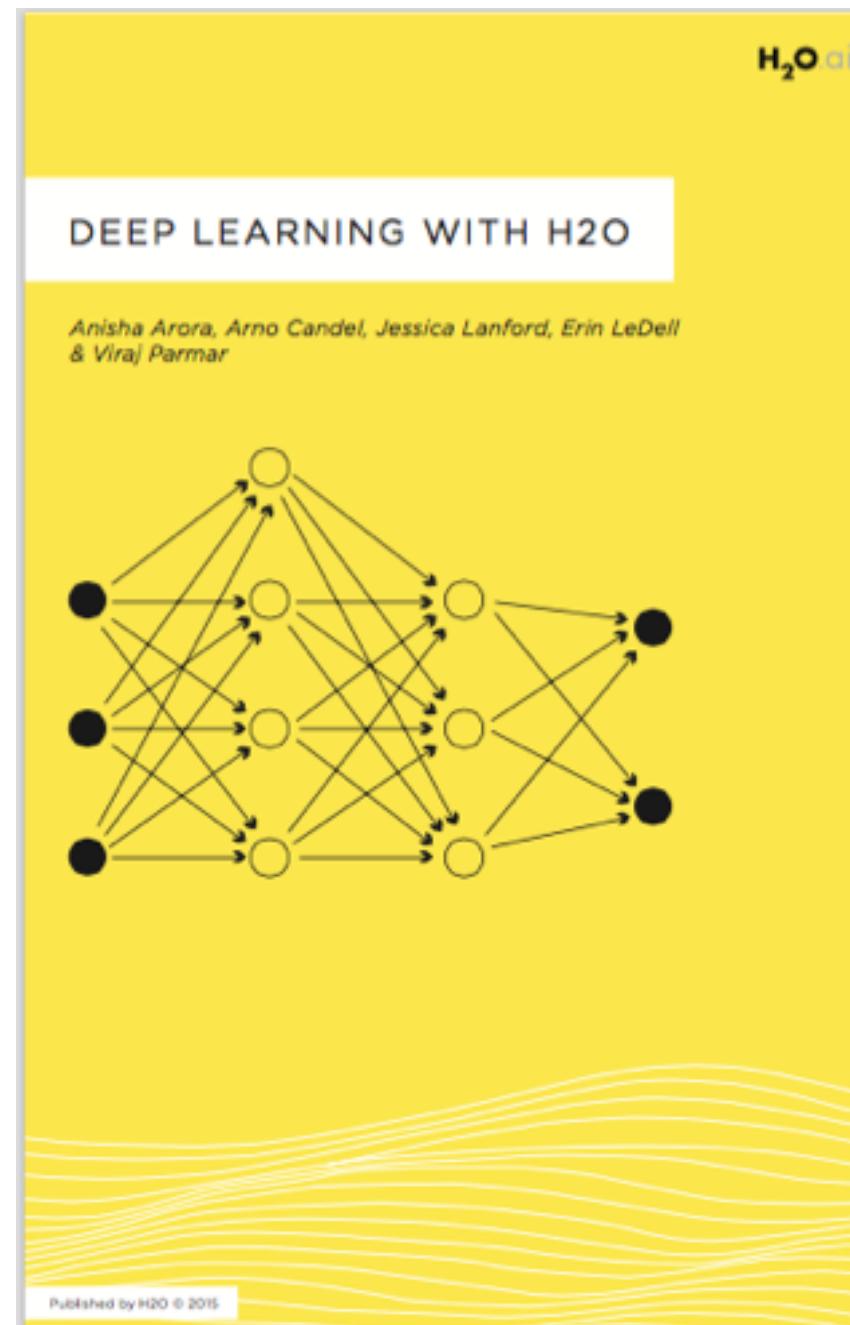
RESOURCES

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>

My Previous H2O Talks



- bit.ly/joe_h2o_talk1
- bit.ly/joe_h2o_talk2
- bit.ly/joe_h2o_talk3
- bit.ly/joe_h2o_talk4
- ...



London
Kaggle

WHAT'S NEXT

What's Next?

- H2O + TensorFlow
- H2O + mxnet
- H2O + GPU
- H2O + x ... y ... z
- ...



h2oai / **sparkling-water**

Code Issues Pull requests Pulse Graphs Settings

Branch: master · **sparkling-water** / py / examples / notebooks / TensorFlowDeepLearning.ipynb Find file Copy path

arnocandel Update TensorFlowDeepLearning.ipynb d3376e8 19 days ago

1 contributor

975 lines (974 sloc) | 34.1 KB Raw Blame History

H2O TensorFlow Deep Learning Demo

You can also follow this [Youtube video](#).

Prerequisites

1. Install TensorFlow from <https://www.tensorflow.org>
2. Download Sparkling Water from <http://www.h2o.ai/download/sparkling-water/spark16>
3. Follow [instructions to setup PySparkling](#) (especially steps 1 and 2)
4. Launch a Jupyter Notebook that connects to PySparkling:

```
cd ~/Downloads  
unzip sparkling-water-1.6.5.zip  
cd sparkling-water-1.6.5  
~/sparkling-water-1.6.5$ IPYTHON_OPTS="notebook" bin/pysparkling
```

5. Point the Notebook to [this file](#) (e.g., download it first, then upload into the Notebook)

Introduction

In this tutorial, we'll build a simple 2-layer deep artificial neural network to classify handwritten digits [MNIST](#). If you are not familiar with these terms, please check out our [Deep Learning Booklet](#).

Any Questions?

- Contact
 - joe@h2o.ai
 - @matlabulous
 - github.com/woobe
- Links (All Slides)
 - github.com/h2oai/h2o-meetups
- H2O in London
 - Coming soon!
 - Meetups
 - Office
 - We're hiring!
 - www.h2o.ai/careers

EXTRA SLIDES

R Example – Random Grid Search

Define search range and criteria

```
# Define a list of hyper-parameters
h_param_gbm <- list(
  max_depth      = seq(5, 10, 1),
  sample_rate    = seq(0.5, 0.8, 0.1),
  col_sample_rate = seq(0.5, 0.8, 0.1)
)

# Search Criteria
search_criteria = list(
  strategy       = "RandomDiscrete", # Use random grid search
  max_runtime_secs = 600             # limit the runtime to 10 minutes
)

# Grid Search
grid <- h2o.grid(
  hyper_params   = h_param_gbm,
  search_criteria = search_criteria,
  algorithm      = "gbm",           # which H2O algorithm to run

  # Standard model parameters
  training_frame  = hex_train,
  validation_frame = hex_valid,
  x               = features,
  y               = "OutcomeType",
```

H2O Grid Details

=====

Grid ID: gbm_grid

Used hyper parameters:

- sample_rate
- max_depth
- col_sample_rate

Number of models: 21

Number of failed models: 0

Best models

Hyper-Parameter Search Summary: ordered by increasing logloss

	sample_rate	max_depth	col_sample_rate	model_ids	logloss
1	0.8	7	0.6	gbm_grid_model_2	0.752684243384921
2	0.8	6	0.5	gbm_grid_model_10	0.75318954895972
3	0.8	8	0.6	gbm_grid_model_4	0.753499220269728
4	0.7	6	0.6	gbm_grid_model_8	0.755403405990038
5	0.7	5	0.5	gbm_grid_model_14	0.756978014232093

	sample_rate	max_depth	col_sample_rate	model_ids	logloss
16	0.8	10	0.8	gbm_grid_model_12	0.765548067940223
17	0.6	9	0.7	gbm_grid_model_5	0.765760036358475
18	0.6	10	0.7	gbm_grid_model_16	0.769422559292015
19	0.5	10	0.8	gbm_grid_model_0	0.770202667369843
20	0.7	10	0.8	gbm_grid_model_17	0.773242423940347
21	0.6	10	0.5	gbm_grid_model_18	0.776840032616653

R Example – Model Stacking

Using `h2o.stack(...)` to combine multiple models

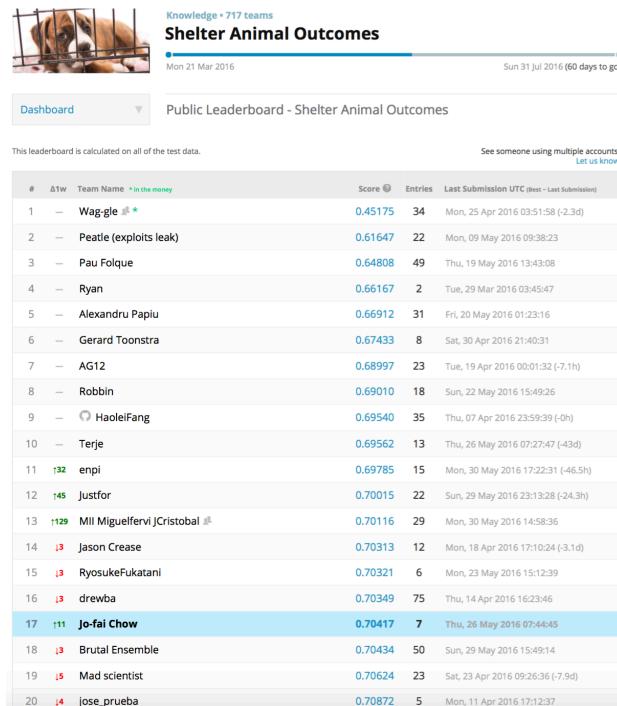
```
# H2O GBM
model_gbm <- h2o.gbm(training_frame = hex_train,
                      x = features, y = target,
                      nfolds = 5,
                      fold_assignment = "Modulo",
                      keep_cross_validation_predictions = TRUE)

# H2O Random Forest
model_drf <- h2o.randomForest(training_frame = hex_train,
                               x = features, y = target,
                               nfolds = 5,
                               fold_assignment = "Modulo",
                               keep_cross_validation_predictions = TRUE)

# Define list of models and metalearner
models <- list(model_gbm, model_drf)
metalearner <- "h2o.glm.wrapper"

# Run h2o.stack for metalearning
stack <- h2o.stack(models = models,
                   response_frame = hex_train[, target],
                   metalearner = metalearner,
                   keep_levelone_data = TRUE)
```

Getting reasonable results



17 out of 717 teams (\approx top 2%)