CHICAGO
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE 2015

goto; conference

Scalable Data Science and Deep Learning with H2O

Arno Candel, PhD
Chief Architect, H2O.ai

Who Am I?

FORTUNE SUBSCRIBE

Arno Candel caught the science bug early. He grew up in Untersiggenthal, Switzerland, a small village wedged between a top particle accelerator lab at the Paul Scherrer Institute and ETH Zürich, continental Europe's most prestigious technical university. Studying particle physics and supercomputing, Candel coded models of the universe on computers. After moving to California to work at the SLAC National Accelerator Laboratory, he moved to the startup world, joining Skytree as a founding engineer and designing high-performance machine learning algorithms. At 0xdata he is a core developer on the data science platform known as h2o, which has been ranked the number one open-source Java machine learning project by members of the coding community GitHub. The platform enables deep learning and is compatible with the popular statistical programming language R. His title at the company? "Physicist & Hacker," of course. — *Robert Hackett*

Arno Candel

Chief Architect,
Physicist & Hacker at H2O.ai

PhD Physics, ETH Zurich 2005
10+ yrs Supercomputing (HPC)
6 yrs at SLAC (Stanford Lin. Accel.)
3.5 yrs Machine Learning
1.5 yrs at H2O.ai

Fortune Magazine
Big Data All Star 2014

Follow me [@ArnoCandel]

Outline

- Introduction
- H2O Deep Learning Architecture
- Live Demos:
  - Flow GUI - Airline Logistic Regression
  - Scoring Engine - Million Songs Classification
  - R - MNIST Unsupervised Anomaly Detection
  - Flow GUI - Higgs Boson Classification
  - Sparkling Water - Chicago Crime Prediction
  - iPython - CitiBike Demand Prediction
- Outlook

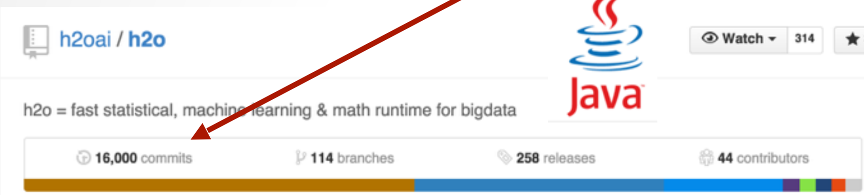
H2O - Product Overview

In-Memory ML	Memory-Efficient Data Structures Cutting-Edge Algorithms
Distributed	Use all your Data (No Sampling) Accuracy with Speed and Scale
Open Source	Ownership of Methods - Apache V2 Easy to Deploy: Bare, Hadoop, Spark, etc.
APIs	Java, Scala, R, Python, JavaScript, JSON NanoFast Scoring Engine (POJO)

Team@H2O.ai

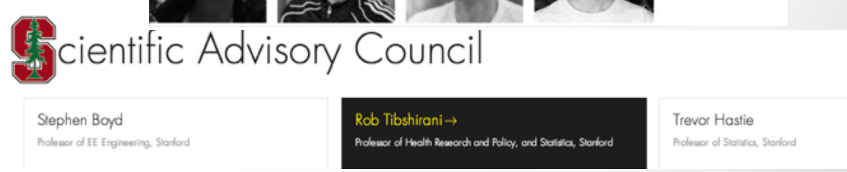
GitHub

25,000 commits / 3yrs



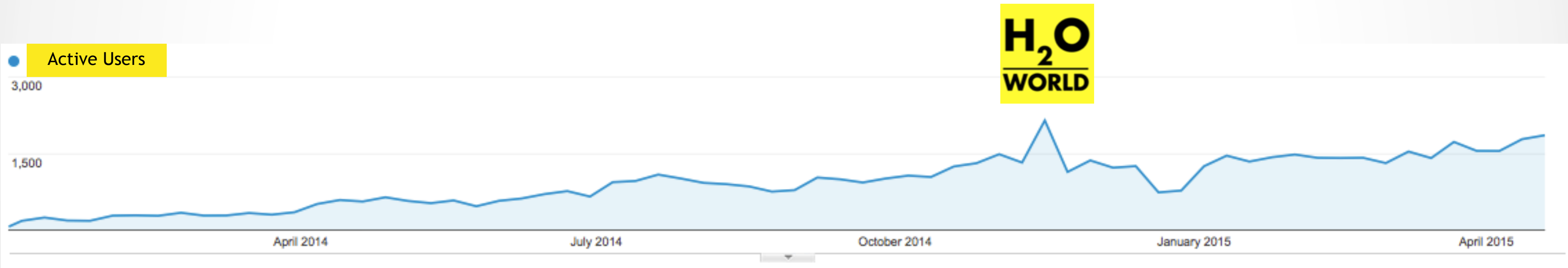
Java

H2O World Conference 2014



Community & Install Base

ML is the new SQL
Prediction is the new Search



150+ Meetup

	Mar 2014	July 2014	Mar 2015
Companies	103	634	2789
Users	463	2,887	13,237

H_2O.ai
Machine Intelligence

Accuracy with Speed and Scale

HDFS

S3

SQL

NoSQL

python™ {JSON} H2O FLOW

Java Scala

Excel tableau

Fast Modeling Engine

Distributed In-Memory

Map Reduce/Fork Join

Columnar Compression

Matrix Factorization →

Feature Engineering

Munging

Supervised

Unsupervised

Clustering →

Classification

Regression

GLM, Deep Learning

K-Means, PCA, NB, Cox

Random Forest / GBM Ensembles

all in-house code

Streaming →

Nano Fast Java Scoring Engines (POJO code generation)

H_2O.ai

Machine Intelligence

Actual Customer Use Cases

Real-time marketing (H2O is 10x faster than anything else)

Ad Optimization (200% CPA Lift with H2O)

P2B Model Factory (60k models, 15x faster with H2O than before)

Fraud Detection (11% higher accuracy with H2O Deep Learning - saves millions)

...and many large insurance and financial services companies!

H2O - Easy Installation

h2o.ai/download

H_2O

Version 0.3.0.1200

The Open Source In-Memory
Prediction Engine for Big Data Science

DOWNLOAD AND RUN | INSTALL IN R | INSTALL IN PYTHON | INSTALL ON HADOOP | USE FROM MAVEN

DOWNLOAD H_2O

Get started with H_2O Dev in 3 easy steps

1. Download H_2O Dev. This is a zip file that contains everything you need to get started.
2. From your terminal, run:

```
cd ~/Downloads
unzip h2o-dev-0.3.0.1200.zip
cd h2o-dev-0.3.0.1200
java -jar h2o.jar
```

3. Point your browser to http://localhost:54321

Help

PACK

examples

- GBM_Example.flow
- DeepLearning_Example.flow
- GLM_Example.flow
- DRF_Example.flow
- K-Means_Example.flow
- Million_Songs.flow

Try it out!

Today's live demos will be run on yesterday's nightly build of the next-gen product (alpha, in QA)!

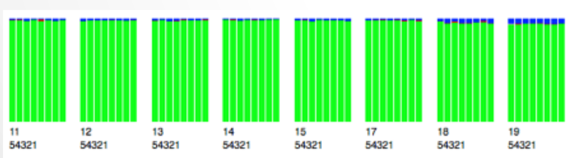
Demo: GLM (Elastic Net) on Airlines

allyears1987to2007.hex

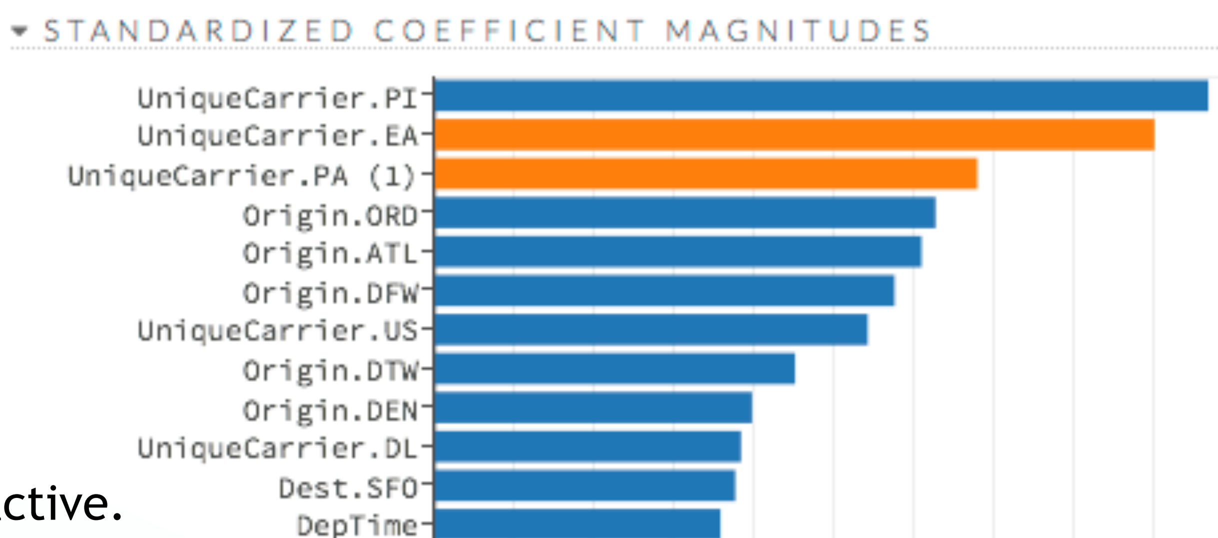
Actions: View Data | Split... | Inspect | Build Model... | Predict | Download

Rows	Columns	Compressed Size
116525241	31	4GB

Description	GLM
Status	DONE
Run Time	00:00:17.485



8 nodes on EC2: all cores active.
Trains on 116M rows in 17 seconds!



Eastern Airlines: Always on time (No schedule!)

POJO Scoring Engine

</> Preview POJO

Standalone Java scoring code is auto-generated!

Example: First GBM tree

```
class gbm_model_Tree_0_class_0 {
  static final double score0(double[] data) {
    double pred =  (data[0 /* C2 */] <45.73866f
      ? (data[13 /* C15 */] <2319.2f
        ? (data[1 /* C3 */] <6.5844727f
          ? (data[0 /* C2 */] <41.95825f
            ? (data[1 /* C3 */] <-31.00589f ? -0.052555863f : -0.09842952f)
            : (data[2 /* C4 */] <14.930695f ? 0.020650635f : -0.049931444f))
          : (data[19 /* C21 */] <532.7656f
            ? (data[56 /* C58 */] <-100.09595f ? -0.07658875f : -0.12227313f)
            : (data[0 /* C2 */] <42.989246f ? -0.094184674f : -0.017104127f)))
        : (data[0 /* C2 */] <40.41174f
          ? (data[13 /* C15 */] <3805.5273f
            ? (data[1 /* C3 */] <-31.617065f ? -0.024733052f : -0.07547661f)
            : (data[3 /* C5 */] <33.312943f ? -0.0056872005f : 0.06485412f))
          : (data[19 /* C21 */] <554.6635f
            ? (data[1 /* C3 */] <-8.562759f ? 0.03362886f : -0.027898893f)
            : (data[5 /* C7 */] <-0.42910385f ? 0.064277075f : 0.021754632f))))
      : (data[19 /* C21 */] <417.1876f
        ? (data[0 /* C2 */] <49.85999f
          ? (data[1 /* C3 */] <24.371841f
            ? (data[24 /* C26 */] <69.74487f ? 0.009623487f : 0.068527505f)
            : (data[56 /* C58 */] <-57.408203f ? 0.02314878f : -0.055622514f))
          : (data[1 /* C3 */] <78.71591f
            ? (data[5 /* C7 */] <-23.057299f ? 0.1039898f : 0.057479307f)
            : (data[0 /* C2 */] <52.319862f ? -0.04428978f : 0.0363179f)))
        : (data[0 /* C2 */] <47.831062f
          ? (data[19 /* C21 */] <552.9827f
            ? (data[1 /* C3 */] <5.254776f ? 0.06647134f : 0.010973259f)
            : (data[58 /* C60 */] <79.95862f ? 0.10148823f : 0.06370963f))
          : (data[40 /* C42 */] <-22.352417f
            ? (data[87 /* C89 */] <0.09623718f ? 0.14305091f : 0.11153427f)
            : (data[19 /* C21 */] <596.9206f ? 0.07924522f : 0.1226203f)))));
    return pred;
  }
}
```

Note:
no heap allocation,
pure decision-making

Help

PACK

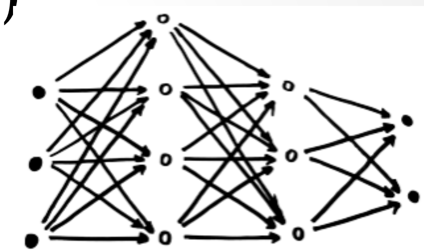
examples

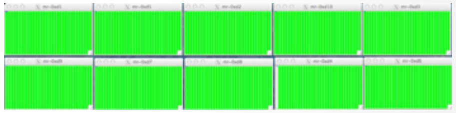
- GBM_Example.flow
- DeepLearning_Example.flow
- GLM_Example.flow
- DRF_Example.flow
- K-Means_Example.flow
- Million_Songs.flow

Fast and easy path to production (batch or real-time)!

H2O Deep Learning

Multi-layer feed-forward Neural Network
Trained with back-propagation (SGD, AdaDelta)





all 320 cores maxed out

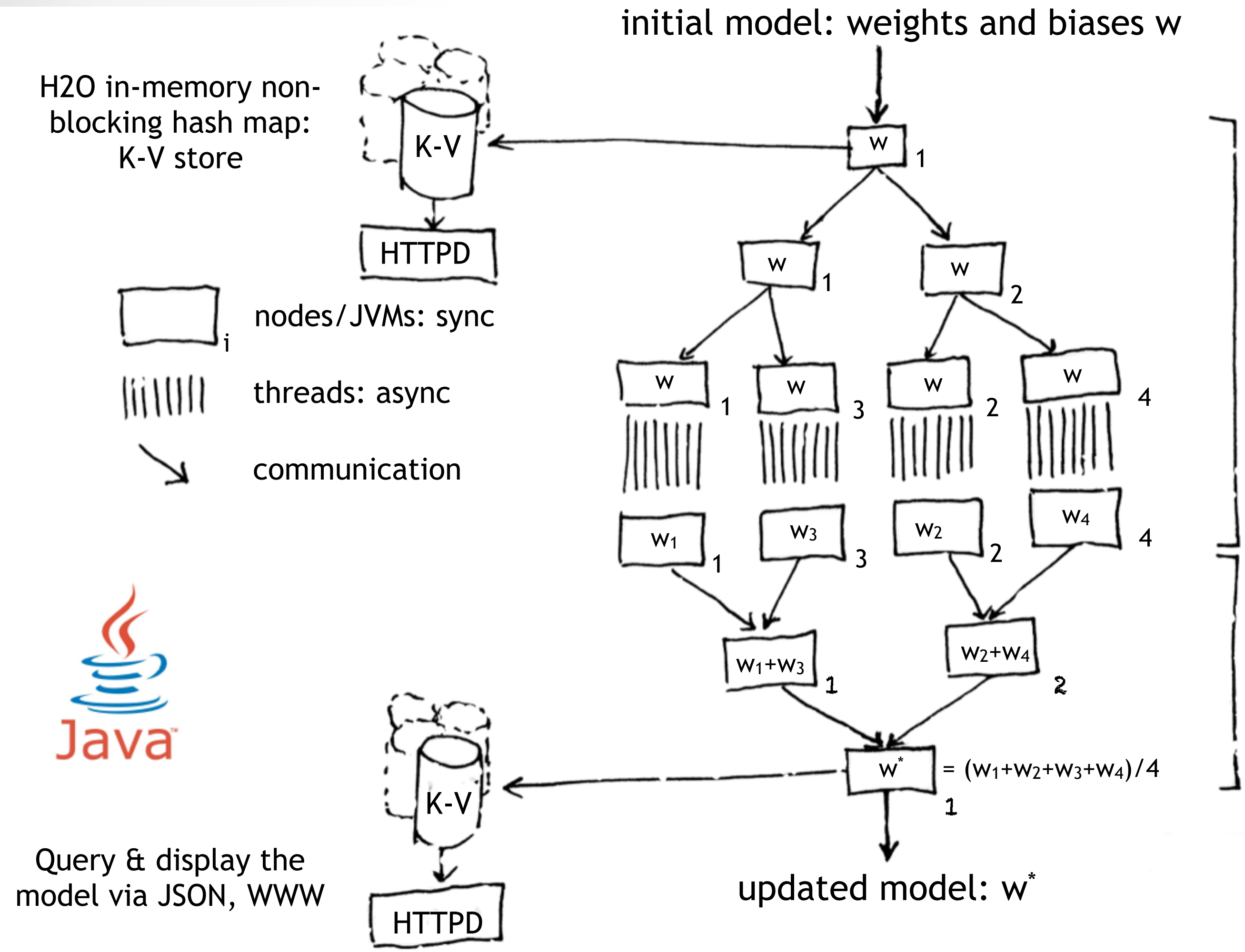
+ distributed processing for big data
  (fine-grain in-memory MapReduce on distributed data)

+ multi-threaded speedup
  (async fork/join worker threads operate at FORTRAN speeds)

+ smart algorithms for fast & accurate results
  (automatic standardization, one-hot encoding of categoricals, missing value imputation, weight & bias initialization, adaptive learning rate, momentum, dropout/l1/L2 regularization, grid search, N-fold cross-validation, checkpointing, load balancing, auto-tuning, model averaging, etc.)

= powerful tool for (un)supervised machine learning on real-world data

H2O Deep Learning Architecture



Main Loop:

map:

each node trains a copy of the weights and biases with (some* or all of) its local data with asynchronous F/J threads

*auto-tuned (default) or user-specified number of rows per MapReduce iteration

reduce:

model averaging: average weights and biases from all nodes, speedup is at least #nodes/log(#rows)
http://arxiv.org/abs/1209.4129

Keep iterating over the data ("epochs"), score at user-given times

H2O Deep Learning beats MNIST

Handwritten digits: 28^2=784 gray-scale pixel values

```
> library(h2o)
> h2oServer <- h2o.init(ip="mr-0xd1", port=53322)
> train_hex <- h2o.importFile(h2oServer, "mnist/train.csv.gz")
> test_hex  <- h2o.importFile(h2oServer, "mnist/test.csv.gz")
> train_hex[,785] <- as.factor(train_hex[,785])
> test_hex[,785] <- as.factor(test_hex[,785])
> dl_model <- h2o.deeplearning(x=c(1:784), y=785, training_frame=train_hex, l1=1e-5,
                              activation="RectifierWithDropout", input_dropout_ratio=0.2,
                              classification_stop=-1,hidden=c(1024,1024,2048), epochs=8000)
|====================================================================================================
===================================| 100%
> h2o.confusionMatrix(dl_model, test_hex)
Confusion Matrix - (vertical: actual; across: predicted):
          0    1    2    3   4   5   6    7   8    9   Error          Rate
0       974    1    1    0   0   0   2    1   1    0 0.00612 =     6 / 980
1         0 1135    0    1   0   0   0    0   0    0 0.00088 =    1 /1,135
2         0    0 1028    0   1   0   0    3   0    0 0.00388 =   4 / 1,032
3         0    0    1 1003   0   0   0    3   2    1 0.00693 =   7 / 1,010
4         0    0    1    0 971   0   4    0   0    6 0.01120 =    11 / 982
5         2    0    0    5   0 882   1    1   1    0 0.01121 =    10 / 892
6         2    3    0    1   1   2 949    0   0    0 0.00939 =     9 / 958
7         1    2    6    0   0   0   0 1019   0    0 0.00875 =   9 / 1,028
8         1    0    1    3   0   4   0    2 960    3 0.01437 =    14 / 974
9         1    2    0    0   4   3   0    2   0  997 0.01189 =  12 / 1,009
Totals  981 1142 1038 1013 977 891 956 1031 964 1007 0.00830 = 83 / 10,000
```

Standard 60k/10k data
No distortions
No convolutions
No unsupervised training
No ensemble

10 hours on 10 16-core servers

World-record!
0.83% test set error

A Classic Benchmark!

Table 1: *Classification error rate comparison: DBN vs. DCN*

DBN [3] (Hinton's)	DBN (MSR's)	DCN (Fine-tuning)	DCN (no Fine-tuning)	Shallow (D)CN (Fine-tuned single layer)
1.20%	1.06%	0.83%	0.95%	1.10%

Unsupervised Anomaly Detection

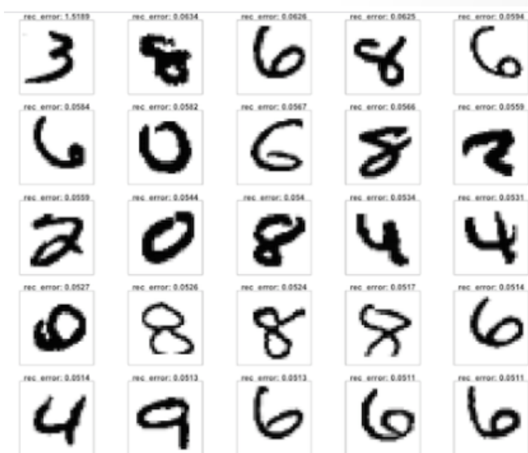
```r
# 1) LEARN WHAT'S NORMAL WITH UNSUPERVISED AUTOENCODER
ae_model <- h2o.deeplearning(x=predictors,
                             training_frame=train_hex,
                             activation="Tanh",
                             autoencoder=T,
                             hidden=c(50),
                             l1=1e-5,
                             ignore_const_cols=F,
                             epochs=1)
# 2) DETECT OUTLIERS
test_rec_error <- as.data.frame(h2o.anomaly(ae_model, test_hex))
```

layer	*units*	*type*
1	784	Input
2	50	Tanh
3	784	Tanh

Learns Compressed Identity Function

The good

The bad



The ugly

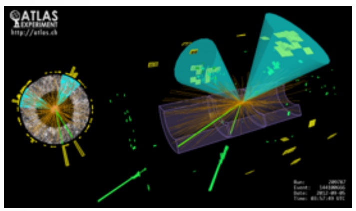
Try it yourself!

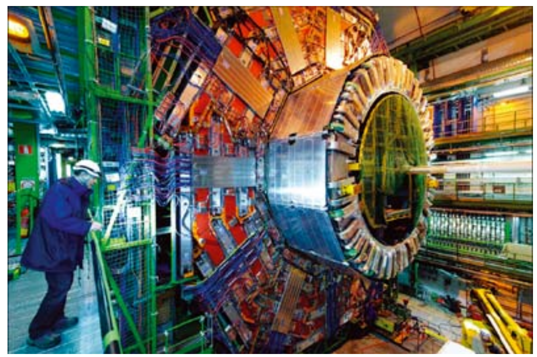
Higgs Boson - Classification Problem

Large Hadron Collider: Largest experiment of mankind!

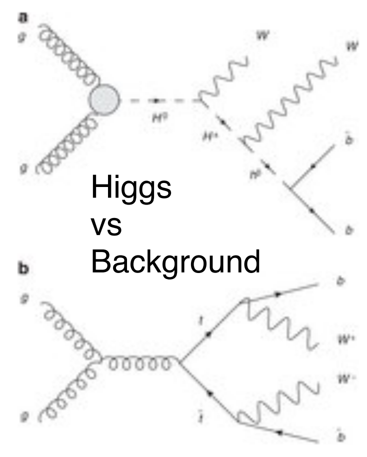
$13+ billion, 16.8 miles long, 120 MegaWatts, -456F, 1PB/day, etc.

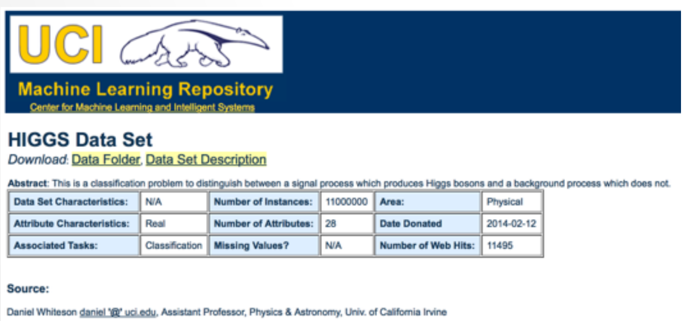
Higgs boson discovery (July '12) led to 2013 Nobel prize!





Images courtesy CERN / LHC





UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

HIGGS Data Set

Download: Data Folder, Data Set Description

Abstract: This is a classification problem to distinguish between a signal process which produces Higgs bosons and a background process which does not.

Data Set Characteristics:	N/A	**Number of Instances:**	11000000	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	28	**Date Donated**	2014-02-12
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	11495

Source:

Daniel Whiteson daniel '@' uci.edu, Assistant Professor, Physics & Astronomy, Univ. of California Irvine

HIGGS UCI Dataset:

21 low-level features AND

7 high-level derived features (physics formulae)

Train: 10M rows, Valid: 500k, Test: 500k rows

Deep Learning for Higgs Detection?

nature.com

Searching for exotic particles in high-energy physics with deep learning

P. Baldi, P. Sadowski & D. Whiteson

Affiliations | Contributions | Corresponding authors

Nature Communications 5, Article number: 4308 | doi:10.1038/ncomms5308
Received 19 February 2014 | Accepted 04 June 2014 | Published 02 July 2014

	AUC		
Technique	Low-level	High-level	Complete
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	?	?	?

Former baseline for AUC: 0.733 and 0.816

H2O Algorithm	low-level H2O AUC	all features H2O AUC
Generalized Linear Model	0.596	0.684
Random Forest	0.764	0.840
Gradient Boosted Trees	0.753	0.839
Neural Net 1 hidden layer	0.760	0.830
H2O Deep Learning	?	

add derived features

H2O.ai
Machine Intelligence

H2O Deep Learning Higgs Demo

HIGGS.hex

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

Rows	Columns	Compressed Size
11000000	29	2GB

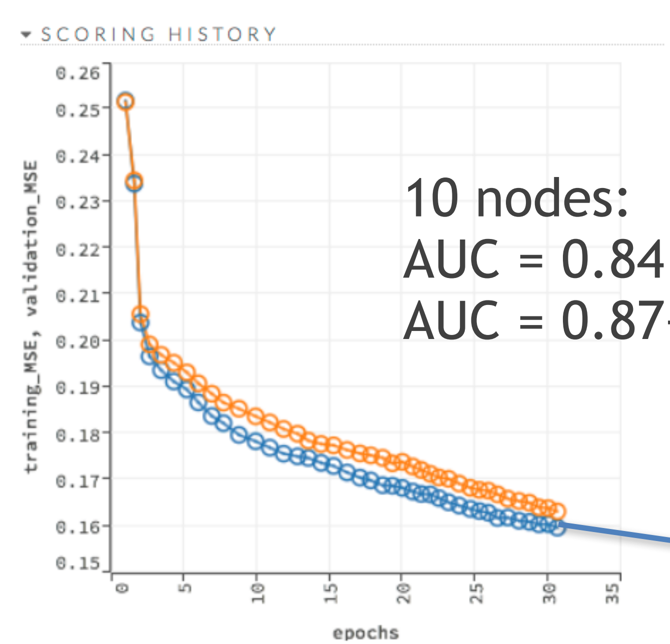
Split Frames

Type	Key	Ratio
	valid	0.05
	test	0.05
	train	0.9

valid 500k rows
test 500k rows
train 10M rows

Deep DL model on 21 low-level features (no physics formula based extra features)

layer	units	type
1	21	Input
2	500	Rectifier
3	500	Rectifier
4	500	Rectifier
5	500	Rectifier
6	2	Softmax



10 nodes:
AUC = 0.84 after 40 mins
AUC = 0.87+ after 8 hours

	AUC		
Technique	Low-level	High-level	Complete
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 ($<$ 0.001)	0.885 (0.002)

Deep Learning learns Physics!

H2O Deep Learning in the News

http://www.datanami.com/2015/05/07/what-police-can-learn-from-deep-learning/

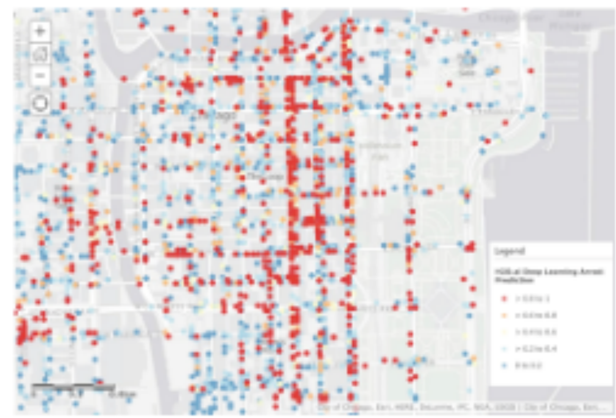
May 7, 2015

What Police Can Learn from Deep Learning

Alex Woodie

Police departments are increasingly turning to predictive analytics to help them fight crime, and the early returns are positive, with double-digit drops in crime rates reported in many cities. So what's next? According to big data analytics experts, police departments could spend their time and money more effectively by giving deep learning algorithms a role in the dispatch room.

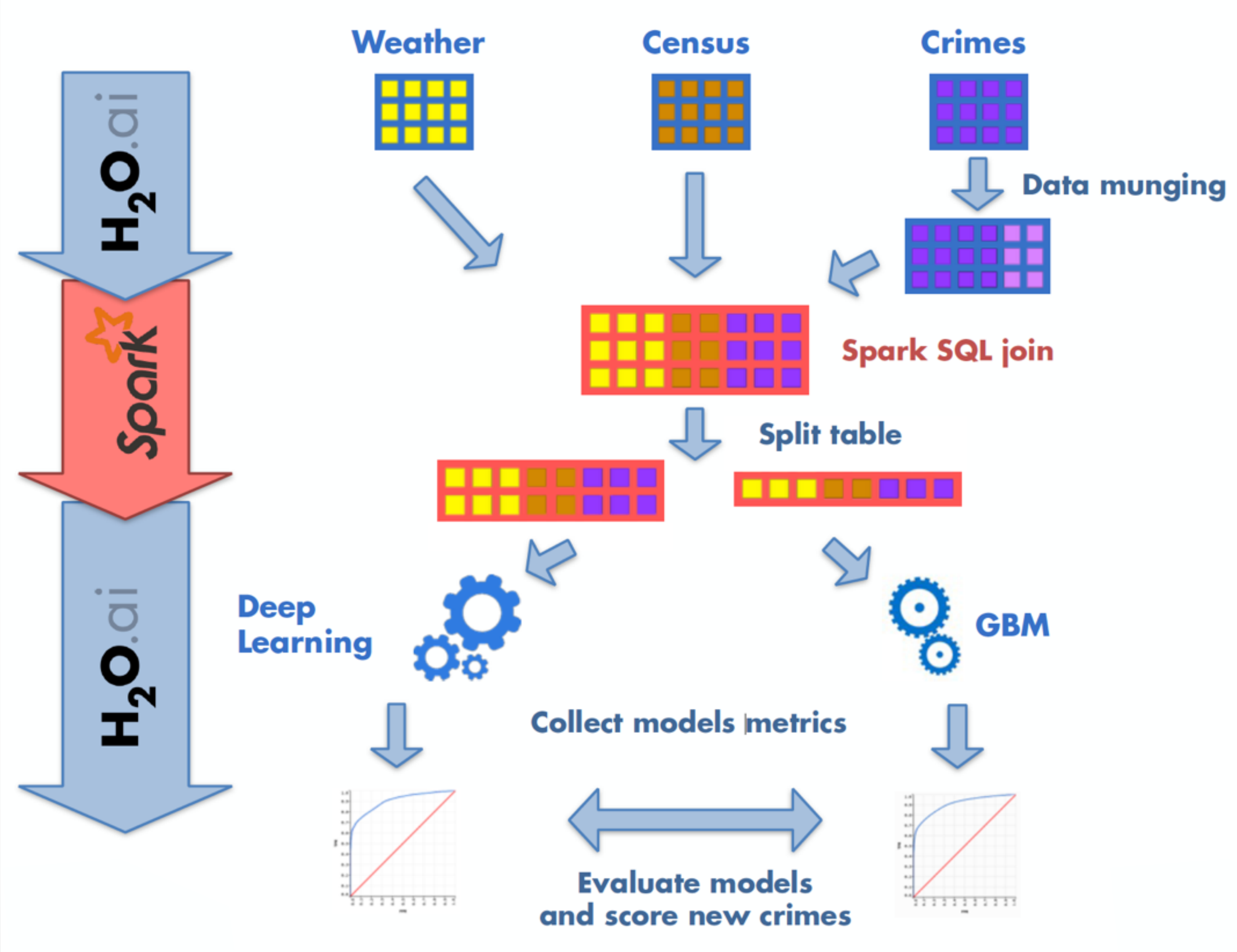
Alex, Michal, et al.



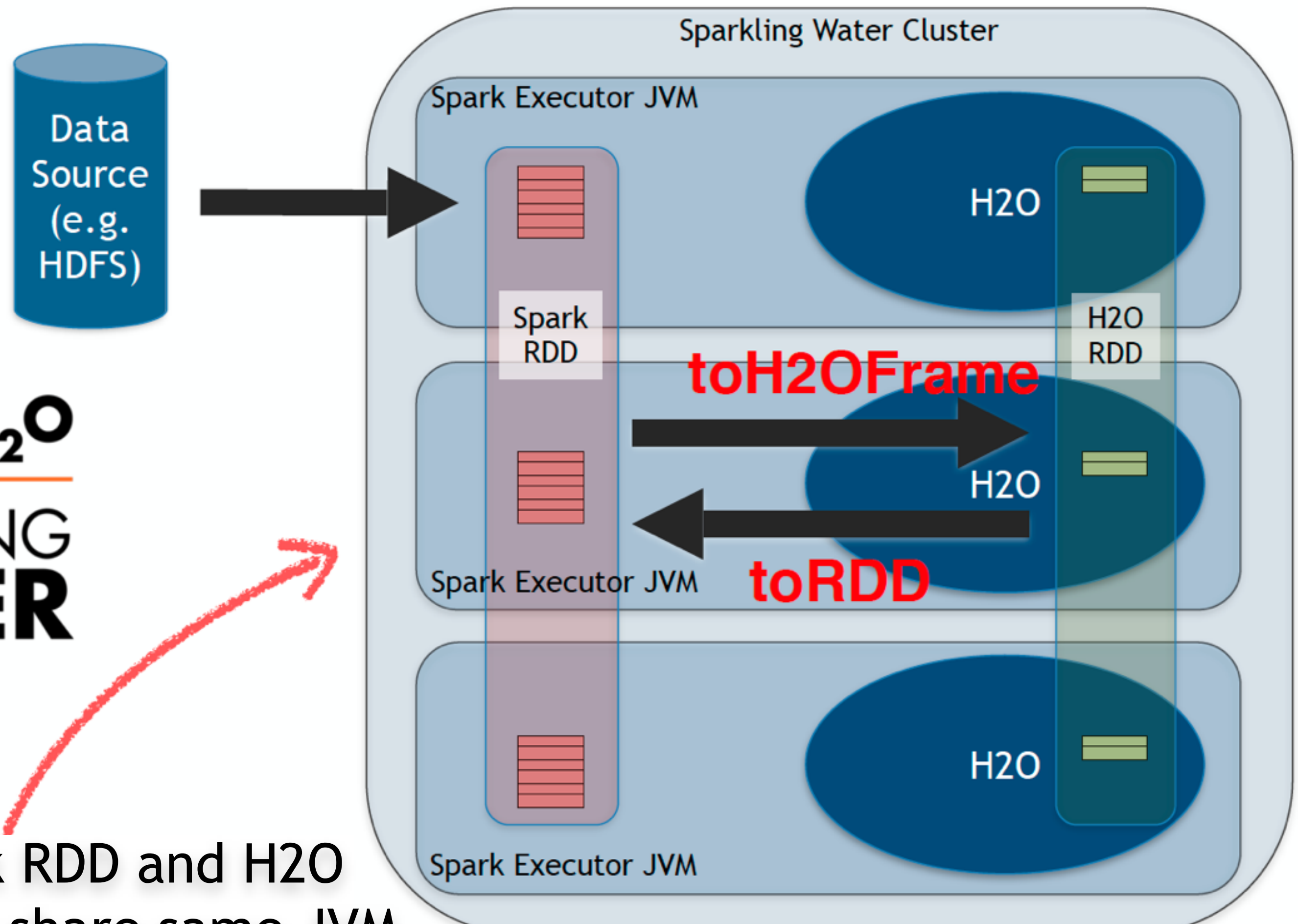
Crimes in Chicago have a geographic element to them that can be serve as an input for deep learning algorithms

http://www.slideshare.net/0xdata/crime-deeplearningkey

Weather + Census + Crime Data



Integration with Spark Ecosystem



Spark RDD and H2O Frames share same JVM

Sparkling Water Demo

branch: **master** sparkling-water / examples / scripts / **chicagoCrimeSmallShell.script.scala**

Instructions at h2o.ai/download

```
/**
 * To start Sparkling Water please type

cd path/to/sparkling/water
export SPARK_HOME="your/spark-1.2.0-installation"
export MASTER="local-cluster[3,2,4096]"

bin/sparkling-shell --conf spark.executor.memory=3G
*/
```

```
//
// Prepare environment
//
import hex.deeplearning.DeepLearningModel
import hex.tree.gbm.GBMModel
import hex.tree.gbm.GBMModel.GBMParameters.Family
import hex.{Model, ModelMetricsBinomial}
import org.apache.spark.SparkFiles
import org.apache.spark.examples.h2o.DemoUtils._
import org.apache.spark.examples.h2o.{Crime, RefineDateColumn}
import org.apache.spark.h2o._
import org.apache.spark.sql._
// SQL support
implicit val sqlContext = new SQLContext(sc)
import sqlContext._
```

Parse & Munge with H2O, Convert to RDD

```
//
// Start H2O services
//
implicit val h2oContext = new H2OContext(sc).start()
import h2oContext._
```

```
//
// H2O Data loader using H2O API
//
def loadData(datafile: String): DataFrame = new DataFrame(new java.net.URI(datafile))

//
// Loader for weather data
//
def createWeatherTable(datafile: String): DataFrame = {
  val table = loadData(datafile)
  // Remove first column since we do not need it
  table.remove(0).remove()
  table.update(null)
  table
}
```

H2O Parser: Robust & Fast

Simple Column Selection

Parse & Munge with H2O, Convert to RDD

```scala
//
// Load and modify crime data
//
def createCrimeTable(datafile: String, datePattern:String, dateTimeZone:String): DataFrame = {
  val table = loadData(datafile)
  // Refine date into multiple columns
  val dateCol = table.vec(2)
  table.add(new RefineDateColumn(datePattern, dateTimeZone).doIt(dateCol))
  // Update names, replace all ' ' by '_'
  val colNames = table.names().map( n => n.trim.replace(' ', '_'))
  table._names = colNames
  // Remove Date column
  table.remove(2).remove()
  // Update in DKV
  table.update(null)
  table
}
```

Munging: Date Manipulations

```scala
//
// Load data
//
addFiles(sc,
  "examples/smalldata/chicagoAllWeather.csv",
  "examples/smalldata/chicagoCensus.csv",
  "examples/smalldata/chicagoCrimes10k.csv"
)

val weatherTable = asSchemaRDD(createWeatherTable(SparkFiles.get("chicagoAllWeather.csv")))
registerRDDAsTable(weatherTable, "chicagoWeather")
// Census data
val censusTable = asSchemaRDD(createCensusTable(SparkFiles.get("chicagoCensus.csv")))
registerRDDAsTable(censusTable, "chicagoCensus")
// Crime data
val crimeTable  = asSchemaRDD(createCrimeTable(SparkFiles.get("chicagoCrimes10k.csv"), "MM/dd/yyyy hh:mm:ss a", "Etc/UTC"))
registerRDDAsTable(crimeTable, "chicagoCrime")
```

Conversion to SchemaRDD

H2O.ai
Machine Intelligence

Join RDDs with SQL, Convert to H2O

Spark SQL Query Execution

```
//
// Join crime data with weather and census tables
//
val crimeWeather = sql(
  """SELECT
    |a.Year, a.Month, a.Day, a.WeekNum, a.HourOfDay, a.Weekend, a.Season, a.WeekDay,
    |a.IUCR, a.Primary_Type, a.Location_Description, a.Community_Area, a.District,
    |a.Arrest, a.Domestic, a.Beat, a.Ward, a.FBI_Code,
    |b.minTemp, b.maxTemp, b.meanTemp,
    |c.PERCENT_AGED_UNDER_18_OR_OVER_64, c.PER_CAPITA_INCOME, c.HARDSHIP_INDEX,
    |c.PERCENT_OF_HOUSING_CROWDED, c.PERCENT_HOUSEHOLDS_BELOW_POVERTY,
    |c.PERCENT_AGED_16__UNEMPLOYED, c.PERCENT_AGED_25__WITHOUT_HIGH_SCHOOL_DIPLOMA
    |FROM chicagoCrime a
    |JOIN chicagoWeather b
    |ON a.Year = b.year AND a.Month = b.month AND a.Day = b.day
    |JOIN chicagoCensus c
    |ON a.Community_Area = c.Community_Area_Number""".stripMargin)
```

```
//
// Publish as H2O Frame
crimeWeather.printSchema()
val crimeWeatherDF:DataFrame = crimeWeather
```

Convert back to H2OFrame

```
//
// Split final data table
//
import org.apache.spark.examples.h2o.DemoUtils._
val keys = Array[String]("train.hex", "test.hex")
val ratios = Array[Double](0.8, 0.2)
val frs = splitFrame(crimeWeatherDF, keys, ratios)
val (train, test) = (frs(0), frs(1))
```

Split into Train 80% / Test 20%

Build H2O Deep Learning Model

```
def DLModel(train: DataFrame, test: DataFrame, response: String)
           (implicit h2oContext: H2OContext) : DeepLearningModel = {
  import h2oContext._
  import hex.deeplearning.DeepLearning
  import hex.deeplearning.DeepLearningModel.DeepLearningParameters

  val dlParams = new DeepLearningParameters()
  dlParams._train = train
  dlParams._valid = test
  dlParams._response_column = response
  dlParams._variable_importances = true
  // Create a job
  val dl = new DeepLearning(dlParams)
  val model = dl.trainModel.get
  model
}
```

```
//
// Build Deep Learning model
//
val dlModel = DLModel(train, test, 'Arrest)
```

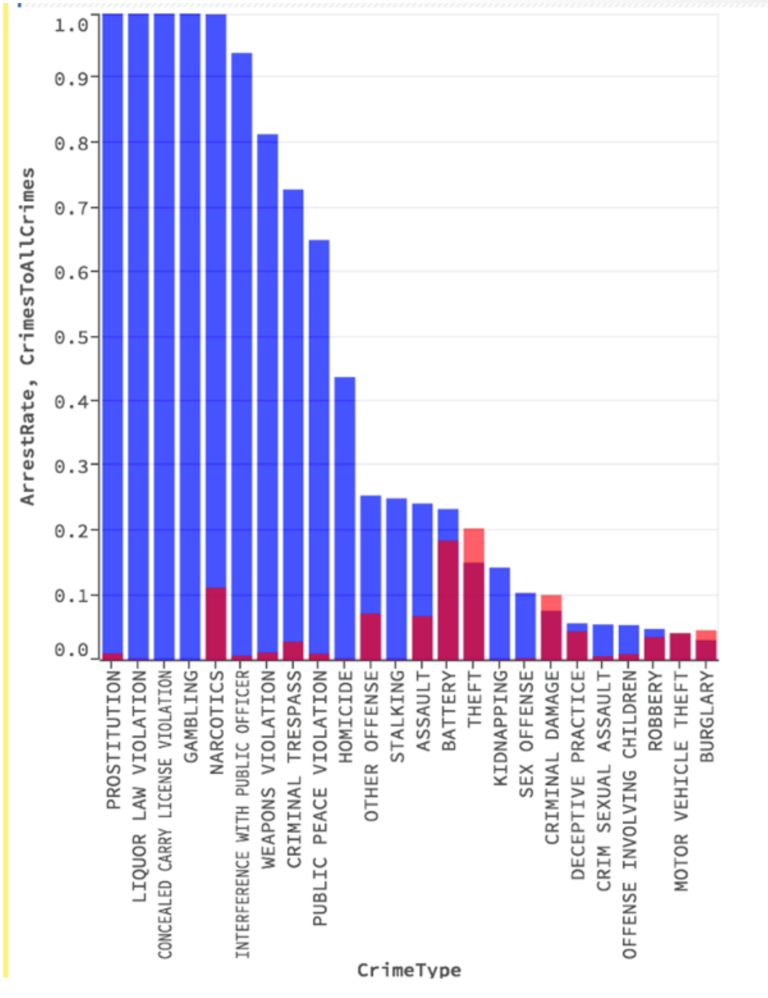
Train a H2O Deep Learning Model on Data obtained by Spark SQL Query

Predict whether Arrest will be made with AUC of 0.90+

Visualize Results with Flow

```
CS
plot (g) -> g(
  g.rect(
    g.position "CrimeType", "ArrestRate"
    g.fillColor g.value 'blue'
    g.fillOpacity g.value 0.75
  )
  g.rect(
    g.position "CrimeType", "CrimesToAllCrimes"
    g.fillColor g.value 'red'
    g.fillOpacity g.value 0.65
  )
  g.from inspect "data", getFrame "frame_rdd_121"
)
```

Using Flow to interactively plot
Arrest Rate (blue)
vs
Relative Occurrence (red)
per crime type.



R's data.table now in H2O!

Package 'data.table'

February 19, 2015

Version 1.9.4

Title Extension of data.frame

Author M Dowle, T Short, S Lianoglou, A Srinivasan with contributions from R Saporta, E Antonyan

Maintainer Matt Dowle <mdowle@mdowle.plus.com>

Freakonometrics

BIG DATA, COMPUTER, DATABASES, MAPS

WORKING WITH "LARGE" DATASETS, WITH DPLYR AND DATA.TABLE

04/05/2015 ARTHUR CHARPENTIER 8 COMMENTS

And here, it took about 2 minutes… The longest part was to extract those first and last observations. So far, it looks like data.table is just perfect to deal with those "large" datasets.

Pinned Tweet

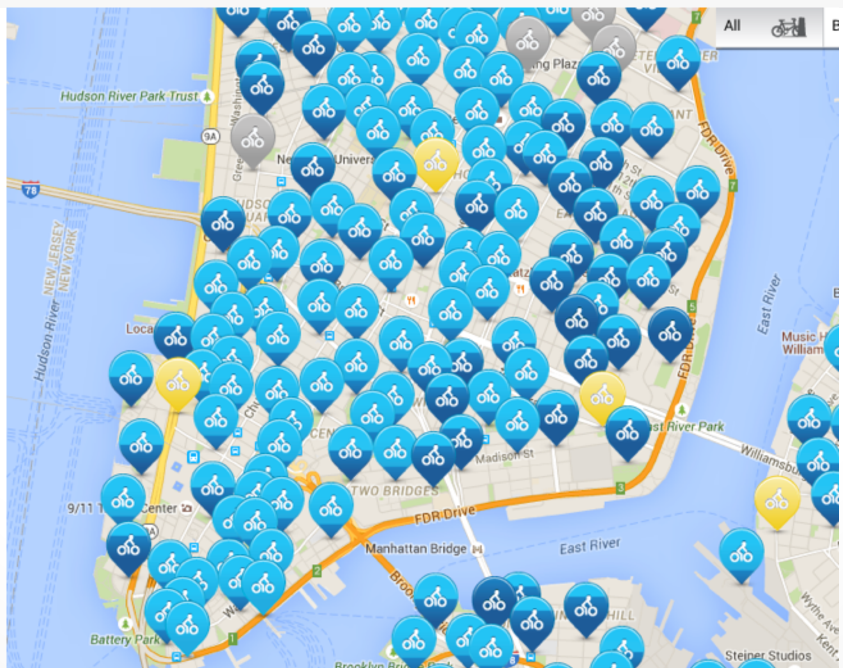
Matt Dowle @MattDowle · Mar 15

Excited to be starting full time at @h2oai in Mountain View tomorrow.

iPython Notebook CitiBike Demo

branch: **master** h2o-dev / h2o-py / demos / **citi_bike_small.ipynb**

In [2]:
```
# Explore a typical Data Science workflow with H2O and Python
#
# Goal: assist the manager of CitiBike of NYC to load-balance the bicycles
# across the CitiBike network of stations, by predicting the number of bike
# trips taken from the station every day.  Use 10 million rows of historical
# data, and eventually add weather data.
```



Cliff et al.

Group-By Aggregation

```
In [5]: # Now do a monster Group-By.  Count bike starts per-station per-day.  Ends up
        # with about 340 stations times 400 days (140,000 rows).  This is what we want
        # to predict.
        by = ["Days","start station name"]
        aggregates = {"bikes": ["count", 0, "all"]}
        bpd = h2o.group_by(data, cols=by, aggregates=aggregates) # Compute bikes-per-day
        bpd.show()
        bpd.describe()
        bpd.dim()
```

Displaying 10 row(s):

Row ID	Days	start station name	bikes
1	[15981.0]	[u'Fulton St & William St']	[89.0]
2	[16007.0]	[u'W 27 St & 7 Ave']	[203.0]
3	[15980.0]	[u'9 Ave & W 18 St']	[137.0]
4	[15995.0]	[u'Washington Ave & Park Ave']	[29.0]
5	[15979.0]	[u'Bedford Ave & S 9th St']	[8.0]
6	[16005.0]	[u'W 13 St & 6 Ave']	[138.0]
7	[15979.0]	[u'11 Ave & W 27 St']	[139.0]
8	[15986.0]	[u'Central Park S & 6 Ave']	[123.0]
9	[16004.0]	[u'John St & William St']	[60.0]
10	[15989.0]	[u'Allen St & Hester St']	[110.0]

Rows: 10,450 Cols: 3

Model Building And Scoring

```
In [9]: # Split the data (into test & train), fit some models and predict on the holdout data
        split_fit_predict(bpd)
        # Here we see an r^2 of 0.91 for GBM, and 0.71 for GLM.  This means given just
        # the station, the month, and the day-of-week we can predict 90% of the
        # variance of the bike-trip-starts.

Training data has 5 columns and 6247 rows, test has 3164 rows, holdout has 1039

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

drf Model Build Progress: [##################################################] 100%

glm Model Build Progress: [##################################################] 100%

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

Model	R2 TRAIN	R2 TEST	R2 HOLDOUT	Model Training Time (s)
GBM	0.997863069083	0.92453193079	0.9058312743	6.732
DRF	0.831504093401	0.786203336569	0.780234326364	5.628
GLM	0.860534716668	0.84755659058	0.833198032239	0.157
DL	0.97405671444	0.92066423657	0.911478042616	6.762

dashboard

91% AUC baseline

Joining Bikes-Per-Day with Weather

```
In [14]: # Lets drop off the extra time columns to make a easy-to-handle dataset.
         wthr4 = wthr3.drop("Year Local").drop("Month Local").drop("Day Local").drop("Hour Local").d
         rop("msec")
```

```
In [15]: # Also, most rain numbers are missing - lets assume those are zero rain days
         rain = wthr4["Rain (mm)"]
         rain[rain == None ] = 0
```

```
In [16]: # ----------
         # 6 - Join the weather data-per-day to the bike-starts-per-day
         print "Merge Daily Weather with Bikes-Per-Day"
         bpd_with_weather = bpd.merge(wthr4,allLeft=True,allRite=False)
         bpd_with_weather.describe()
         bpd_with_weather.show()
```

```
Merge Daily Weather with Bikes-Per-Day
Rows: 10,450 Cols: 10

Chunk compression summary:
```

chunk_type	count	count_percentage	size	size_percentage
CBS	32	10.0	3.5 KB	0.9323617
C1	32	10.0	12.3 KB	3.3141892
C1N	32	10.0	12.3 KB	3.3141892
C1S	32	10.0	12.8 KB	3.4485836
C2	64	20.0	45.1 KB	12.114404
CXD	20	6.25	3.3 KB	0.8845887
C8D	108	33.75	282.7 KB	75.991684

Data Compression Summary:

bitset 0/1

signed byte -128..127

unsigned byte 0..255

1 byte floating point (e.g., 0.493..0.684)

short integers (2 bytes)

sparse doubles

dense double

H2O.ai
Machine Intelligence

Improved Models with Weather Data

```
In [17]: # 7 - Test/Train split again, model build again, this time with weather
         split_fit_predict(bpd_with_weather)

Training data has 10 columns and 6284 rows, test has 3142 rows, holdout has 1012

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

drf Model Build Progress: [##################################################] 100%

glm Model Build Progress: [##################################################] 100%

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

Model	R2 TRAIN	R2 TEST	R2 HOLDOUT	Model Training Time (s)
GBM	0.997835338379	0.922614027693	0.929365255664	7.827
DRF	0.895968570144	0.811939811949	0.799997019098	8.916
GLM	0.843764627796	0.840765558156	0.841326120337	0.146
DL	0.96673892961	0.91917621918	0.924742944402	5.62

93% AUC after joining bike and weather data

H2O.ai Machine Intelligence

More Info in H2O Booklets

Deep Learning Booklet
H2O.ai

Gradient Boosted Models with H2O's R Package
H2O.ai

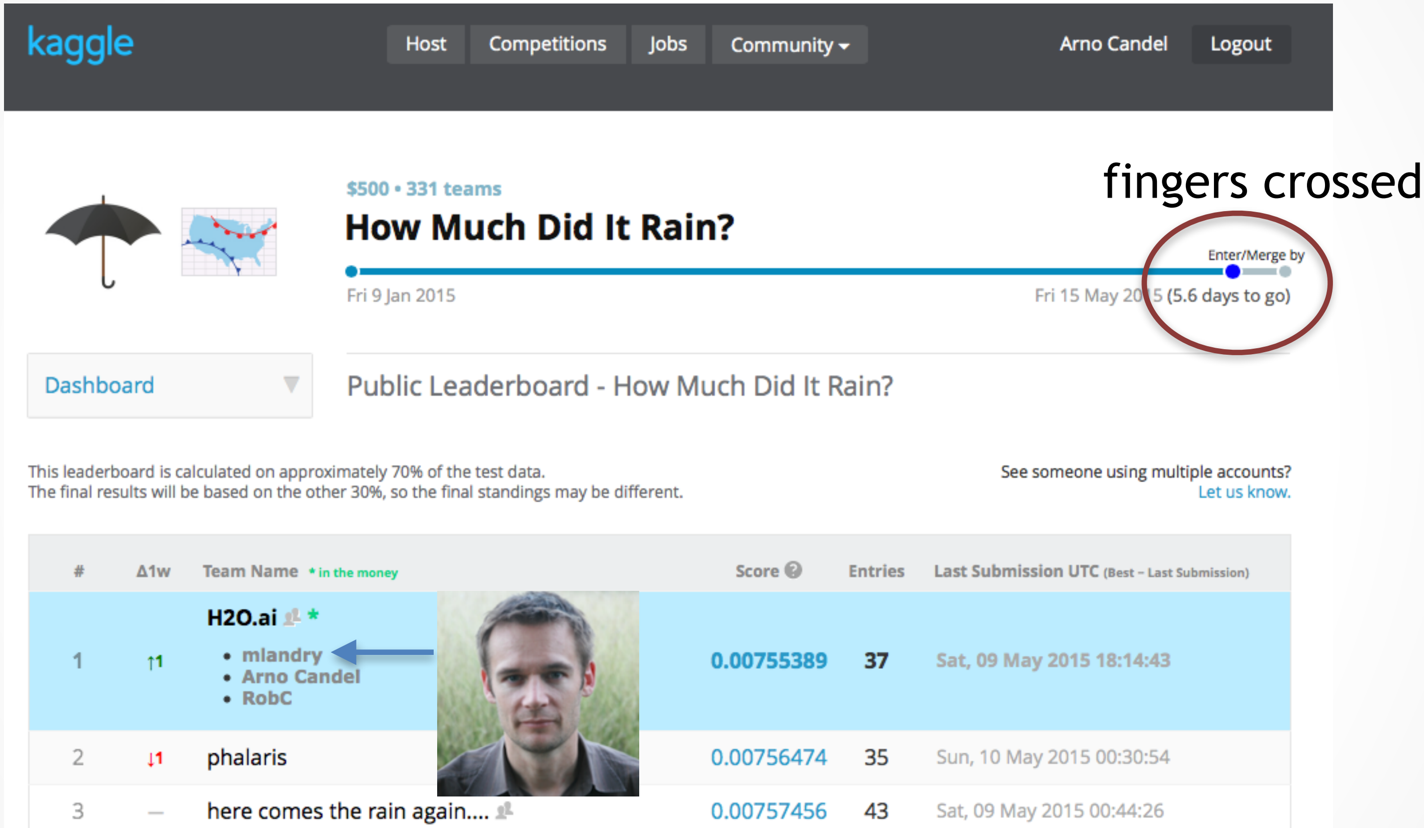
Generalized Linear Modeling with H2O's R Package
H2O.ai

Fast Scalable R with H2O
H2O.ai

https://leanpub.com/u/h2oai

http://learn.h2o.ai

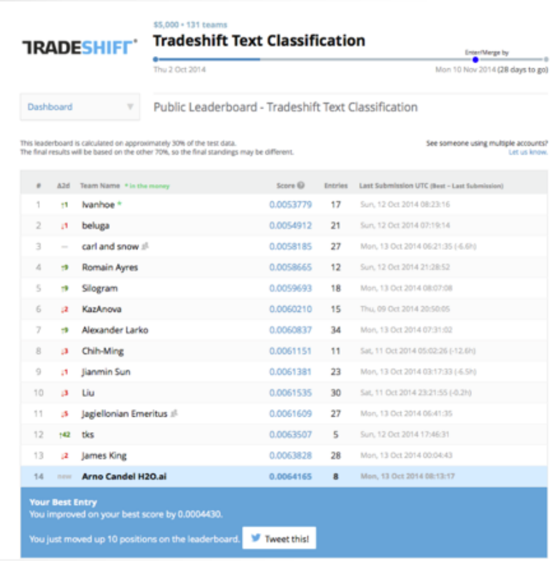
Competitive Data Science

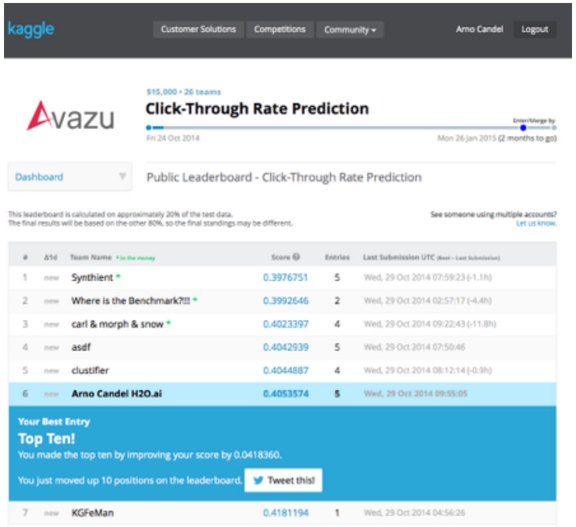


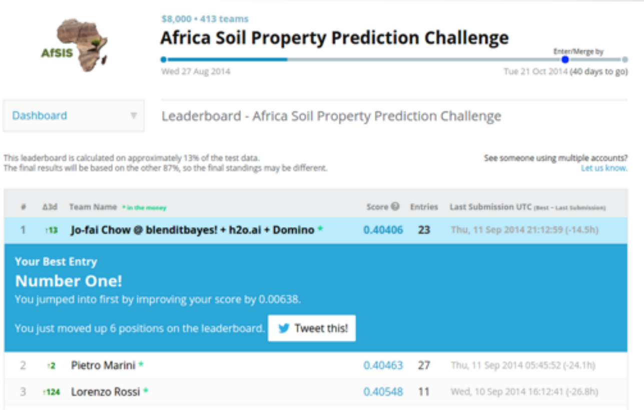
Mark Landry (joined H2O!) will hold a master class on May 19!

http://www.meetup.com/Silicon-Valley-Big-Data-Science/events/222303884/

Past Kaggle Starter Scripts







Higgs challenge

Completed • $13,000 • 1,785 teams

Higgs Boson Machine Learning Challenge

Mon 12 May 2014 – Mon 15 Sep 2014 (59 days ago)

criteo labs

Completed • $16,000 • 718 teams

Display Advertising Challenge

Tue 24 Jun 2014 – Tue 23 Sep 2014 (51 days ago)

Liberty Mutual. INSURANCE

Completed • $25,000 • 634 teams

Liberty Mutual Group - Fire Peril Loss Cost

Tue 8 Jul 2014 – Tue 2 Sep 2014 (2 months ago)

otto group

$10,000 • 3,590 teams

Otto Group Product Classification Challenge

Enter/Merge by

Tue 17 Mar 2015 Mon 18 May 2015 (6.8 days to go)

still ongoing!

Hyper-Parameter Tuning

```r
#########################################################################
## Step 5 - GBM Hyper-Parameter Tuning with Random Search
#########################################################################

models <- c()
for (i in 1:10) {
  rand_numtrees <- sample(1:50,1) ## 1 to 50 trees
  rand_max_depth <- sample(5:15,1) ## 5 to 15 max depth
  rand_min_rows <- sample(1:10,1) ## 1 to 10 min rows
  rand_learn_rate <- 0.025*sample(1:10,1) ## 0.025 to 0.25 learning rate
  model_name <- paste0("GBMModel_",i,
                       "_ntrees",rand_numtrees,
                       "_maxdepth",rand_max_depth,
                       "_minrows",rand_min_rows,
                       "_learnrate",rand_learn_rate
                       )
  model <- h2o.gbm(x=predictors,
                   y=response,
                   training_frame=train_holdout.hex,
                   validation_frame=valid_holdout.hex,
                   destination_key=model_name,
                   loss="multinomial",
                   ntrees=rand_numtrees,
                   max_depth=rand_max_depth,
                   min_rows=rand_min_rows,
                   learn_rate=rand_learn_rate
                   )
  models <- c(models, model)
}
```

Classify products into the correct category

The Otto Group is one of the world's biggest e-commerce companies, with subsidiaries in more than 20 countries, including Crate & Barrel (USA), Otto.de (Germany) and 3 Suisses (France). We are selling millions of products worldwide every day, with several thousand products being added to our product line.

A consistent analysis of the performance of our products is crucial. However, due to our diverse global infrastructure, many identical products get classified differently. Therefore, the quality of our product analysis depends heavily on the ability to accurately cluster similar products. The better the classification, the more insights we can generate about our product range.

93 numerical features

9 output classes

62k training set rows

144k test set rows

h2oai / **h2o-dev**

branch: **master** h2o-dev / h2o-r / demos / **kaggle** / +

Outlook - Algorithm Roadmap

- Ensembles (Erin LeDell et al.)
- Automatic Hyper-Parameter Tuning
- Convolutional Layers for Deep Learning
- Natural Language Processing: tf-idf, Word2Vec
- Generalized Low Rank Models
  - PCA, SVD, K-Means, Matrix Factorization
- Recommender Systems

And many more!

Public JIRAs - Join H2O!

Key Take-Aways

H2O is an open source predictive analytics platform for data scientists and business analysts who need scalable, fast and accurate machine learning.

H2O Deep Learning is ready to take your advanced analytics to the next level.

Try it on your data!

https://github.com/h2oai
H2O Google Group
http://h2o.ai
@h2oai

Thank You!

Questions?

Please remember to evaluate via the GOTO Guide App