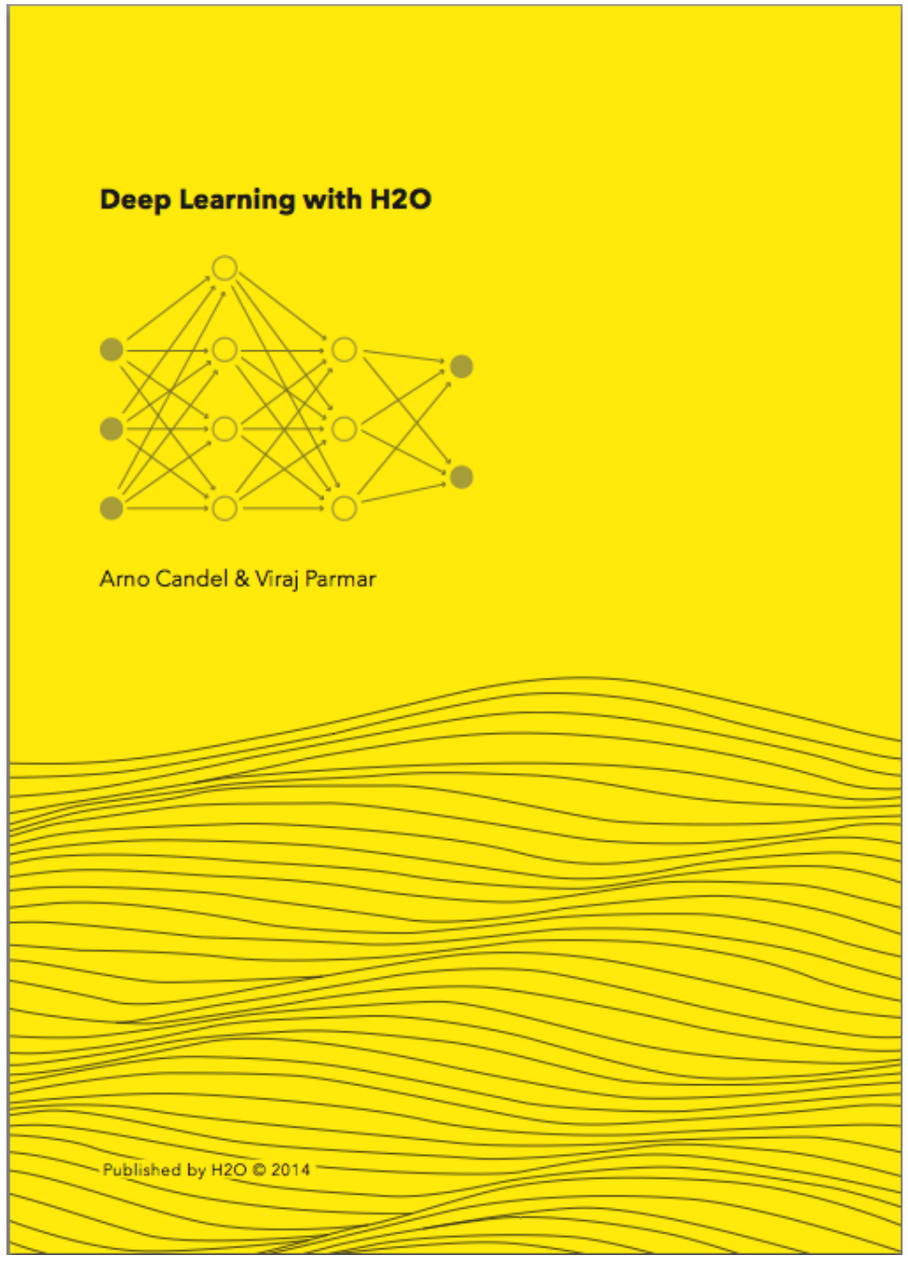
H2O.ai

How to win data science competitions with Deep Learning

@ArnoCandel

Silicon Valley Big Data Science Meetup
0xdata Campus, Mountain View, Oct 9 2014

H2O Deep Learning Booklet!



http://bit.ly/deeplearningh2o

Contents

Who am I?

@ArnoCandel

PhD in Computational Physics, 2005
from ETH Zurich Switzerland

6 years at SLAC - Accelerator Physics Modeling
2 years at Skytree, Inc - Machine Learning
10 months at 0xdata/H2O - Machine Learning

15 years in HPC/Supercomputing/Modeling

Named "2014 Big Data All-Star" by Fortune Magazine

Outline

- **Intro**
  - H2O Architecture
  - Latest News

- **Live Modeling on Kaggle Datasets**
  - African Soil Property Prediction Challenge
  - Display Advertising Challenge
  - Higgs Boson Machine Learning Challenge

- **Hands-On Training - See slides, booklet, H2O.ai**
  - Implementation Concepts
  - Tips & Tricks

- **Outlook**

About H2O (aka 0xdata)

Java, Apache v2 Open Source
Join the www.h2o.ai/community!
#1 Java Machine Learning in Github

GitHub | This repository | Search or type a command | Explore | Features | Enterprise | Blog

PUBLIC 0xdata / h2o

h2o = fast statistical, machine learning & math runtime for bigdata

10,000+ commits | 79 branches | 3 releases | 29 contributors

branch: master | h2o /

Advisors

Systems, Data, File Systems and Hadoop

Doug Lea
ACM Fellow, Malloc for C. fork-join. java memory model, suny oswego

Chris Pouliot
VP Of Data Science, Lyft, formerly, Netflix, Google

Dhruba Borthakur
HDFS, Hive, Facebook

Scientific Advisory Council

Stephen Boyd
Professor of EE Engineering, Stanford

Rob Tibshirani
Professor of Health Research and Policy, and Statistics, Stanford

Trevor Hastie
Professor of Statistics, Stanford

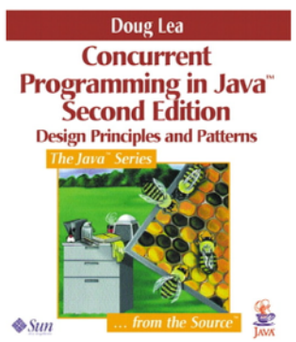
Investors

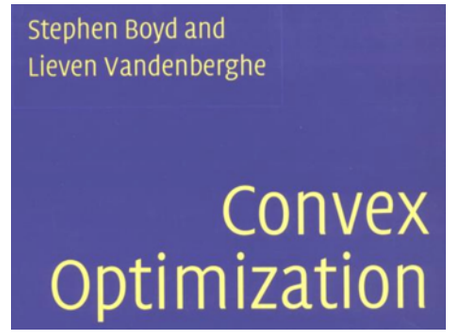
Jishnu Bhattacharjee
Nexus Venture Partners

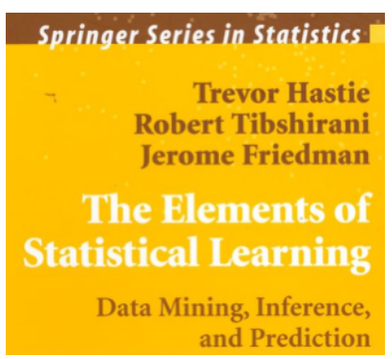
Anand Babu Periasamy
Founder, Gluster (RedHat)

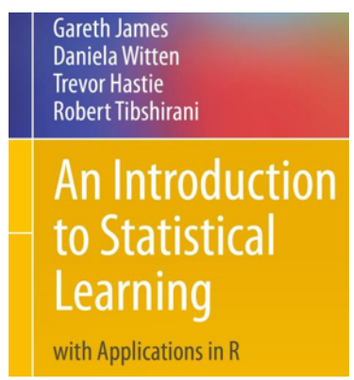
Anand Rajaraman
Founder Junglee (Amazon), Kosmix (WalmartLabs)

Dipchand "Deep" Nishar
SVP of Products & UX (LinkedIn)

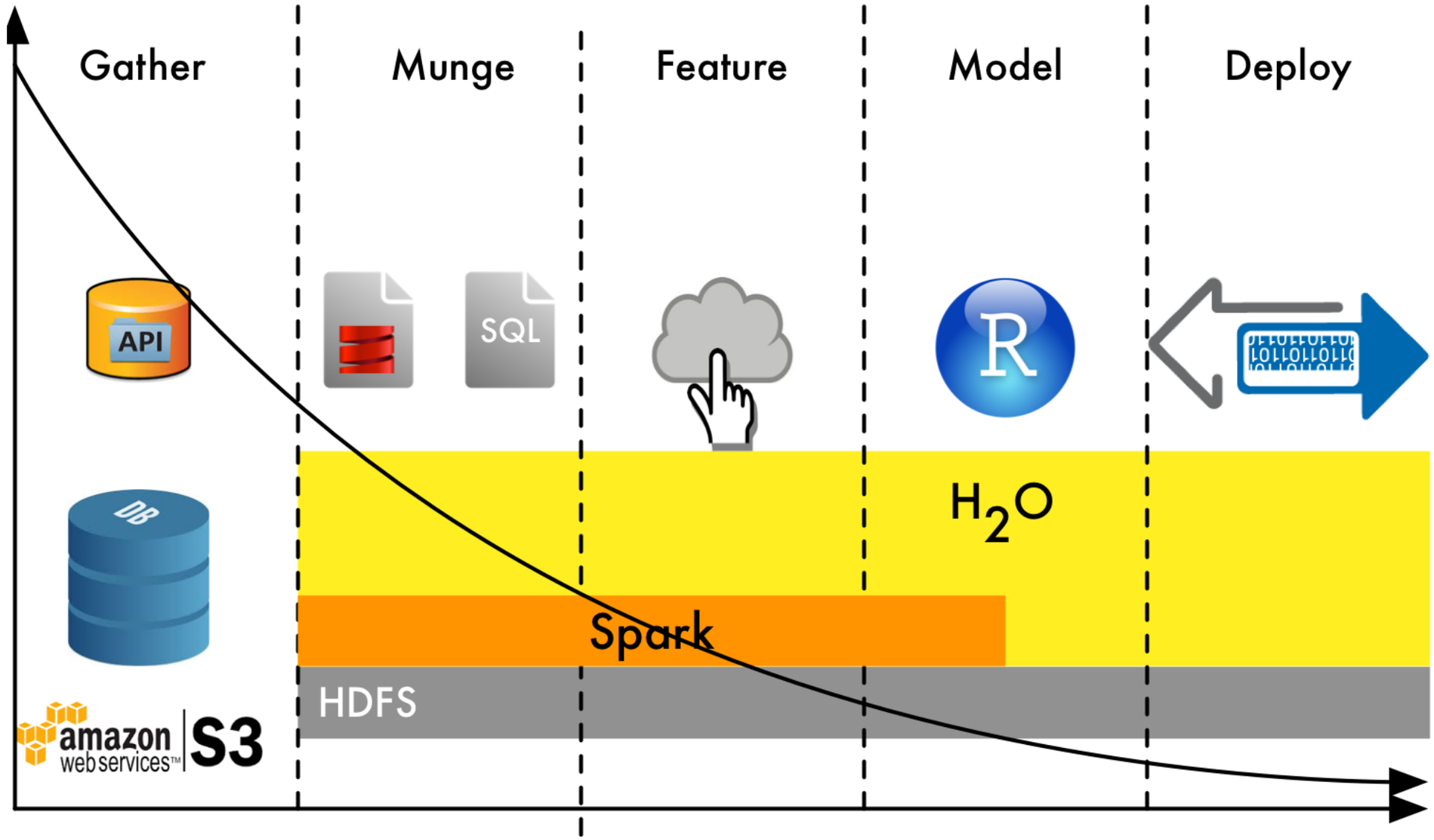




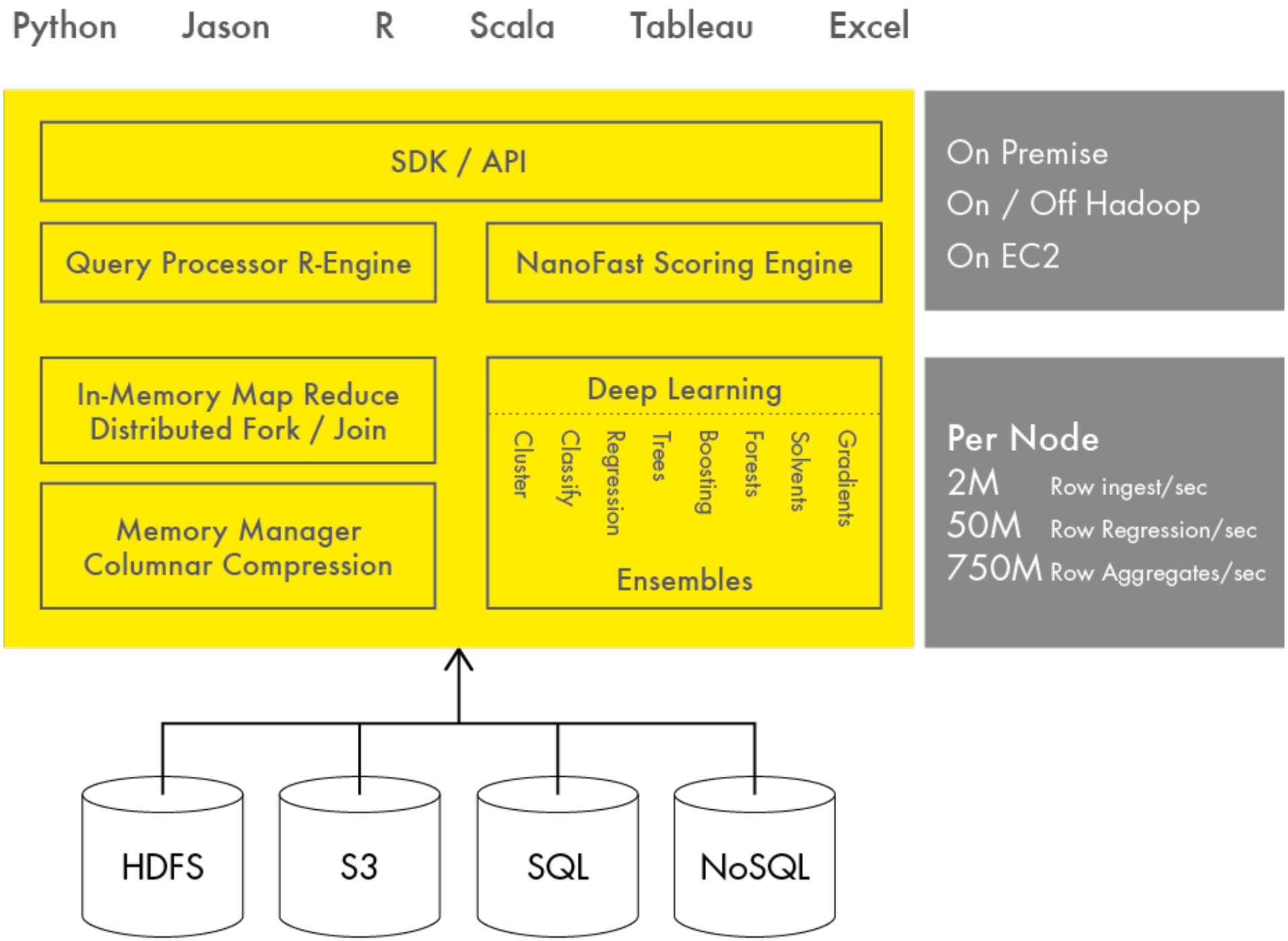




Data Science Workflow



Powering Intelligent Applications



Sparkling Water: Spark + H2O

H2O Announces Sparkling Water - The Killer App for Apache Spark

BusinessWire

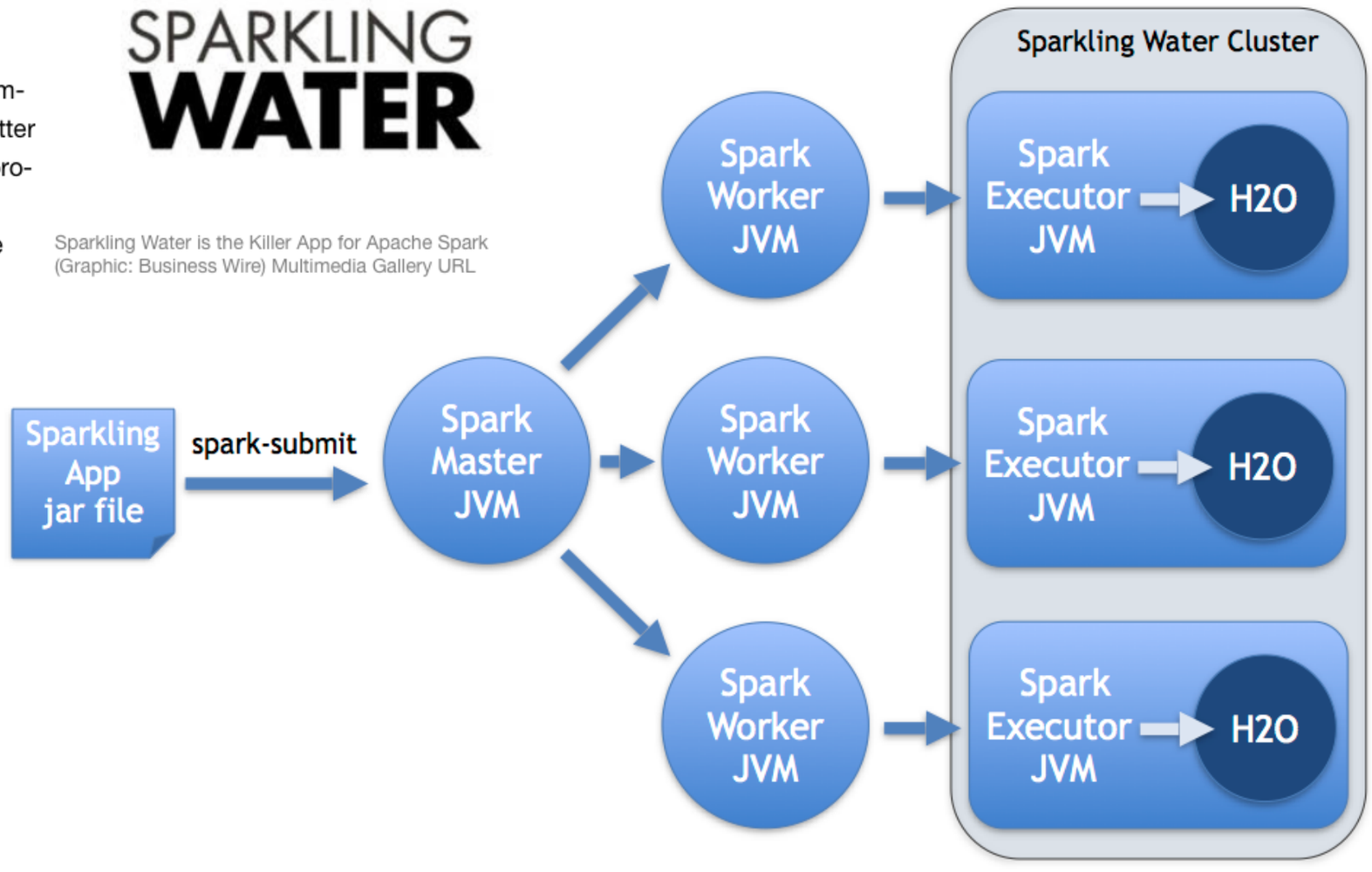
H2O
22 hours ago

SAN FRANCISCO--(BUSINESS WIRE)--

H2O announced today the introduction of Sparkling Water, the latest innovation to combine two best-of-breed open source technologies Apache Spark and H2O. Sparkling Water is the newest application on the Apache Spark in-memory platform to extend Machine Learning for better predictions and to quickly deploy models into production. H2O is proud to partner with Cloudera and Databricks to bring this capability to a wide audience.

Spark + H2O
SPARKLING WATER

Sparkling Water is the Killer App for Apache Spark
(Graphic: Business Wire) Multimedia Gallery URL



"We're thrilled to have H2O bring their machine learning know-how to Apache Spark in the form of Sparkling Water, and look forward to more future collaboration."

–Ion Stoica (CEO, Databricks)

Live Modeling #1

Kaggle
African Soil Properties
Prediction Challenge

kaggle | Customer Solutions | Competitions | Community | Arno Candel H2O.ai | Logout

$8,000 • 1,115 teams

Africa Soil Property Prediction Challenge

Enter/Merge by

Wed 27 Aug 2014 — Tue 21 Oct 2014 (12 days to go)

Still ongoing!

Data fields

SOC, pH, Ca, P, Sand are the five target variables for predictions. The data have been monotonously transformed from the original measurements and thus include negative values.

- **PIDN:** unique soil sample identifier
- **SOC:** Soil organic carbon
- **pH:** pH values
- **Ca:** Mehlich-3 extractable Calcium
- **P:** Mehlich-3 extractable Phosphorus
- **Sand:** Sand content
- **m7497.96 - m599.76**: There are 3,578 mid-infrared absorbance measurements. For example, the "m7497.96" column is the absorbance at wavenumber 7497.96 cm-1. We suggest you to remove spectra CO2 bands which are in the region m2379.76 to m2352.76, but you do not have to.
- **Depth:** Depth of the soil sample (2 categories: "Topsoil", "Subsoil")

Data Files

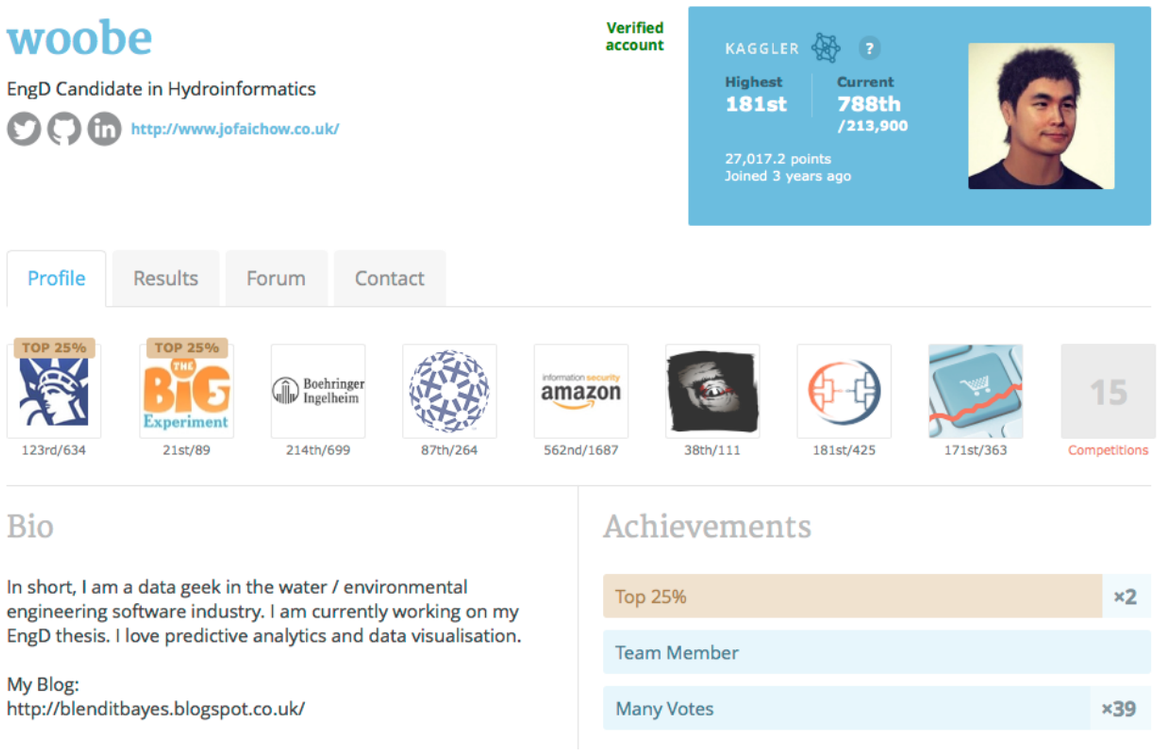
File Name	Available Formats
sample_submission	.csv (13.51 kb)
test	.zip (8.02 mb)
train	.zip (12.67 mb)

File descriptions

- **train.csv** - the training set has 1158 rows.
- **test.csv** - the test set has 728 rows.
- **sample_submission.csv** - all zeros prediction, serving as a sample submission file in the correct format.

Any thoughts? Questions? Challenges? Approaches?

Special thanks to Jo-fai Chow for pointing us to this challenge and for his awesome blog!



How to use R, H2O, and Domino for a Kaggle competition

Guest post by Jo-Fai Chow

The sample project (code and data) described below is available on Domino.

If you're in a hurry, feel free to skip to:

- Tutorial 1: Using Domino
- Tutorial 2: Using H2O to Predict Soil Properties
- Tutorial 3: Scaling up your analysis

Let's follow this tutorial!

Introduction

This blog post is the sequel to TTTAR1 a.k.a. An Introduction to H2O Deep Learning. If the previous blog post was a brief intro, this post is a proper machine learning case study based on a recent Kaggle competition: I am leveraging R, H2O and Domino to compete (and do pretty well) in a real-world data mining contest.

Jo-fai's blog continued

Step 2.2 - Initiate and Connect to a Local H2O Cluster

Using the argument `max_mem_size` in function `h2o.init(...)`, we can manually limit the physical memory usage. In this tutorial, we set a limit of 1GB of memory as it is the maximum allowance of the Domino free tier.

```
library(h2o)
localH2O <- h2o.init(max_mem_size = '1g')
```

launch H2O from R (on your laptop)

```
Successfully connected to http://127.0.0.1:54321
R is connected to H2O cluster:
    H2O cluster uptime:         8 seconds 881 milliseconds
    H2O cluster version:        2.7.0.1510
    H2O cluster name:           H2O_started_from_R
    H2O cluster total nodes:    1
    H2O cluster total memory:   0.97 GB
    H2O cluster total cores:    1
    H2O cluster allowed cores:  15
    H2O cluster healthy:        TRUE
```

single node mode (here)

multi-threading enabled

Step 2.3 - Import Data

Importing data is one of the things I like most about H2O!!! You can load compressed files (`.zip` , `.gz`) directly into the H2O cluster. This may not be significantly important for the small Kaggle dataset in this tutorial but it is certainly useful for bigger datasets.

```
## Import Data to H2O Cluster
train_hex <- h2o.importFile(localH2O, "./data/train.zip")
test_hex <- h2o.importFile(localH2O, "./data/test.zip")
```

import (zipped) data

Jo-fai's blog continued

Step 2.4 - Train Deep Neural Networks Models and Make Predictions

Without further ado, here is the starter code to train one deep neural networks model for each target variable and then use the models to make predictions. There are quite a few variables to tweak in the `h2o.deeplearning(...)` function. To keep things simple, I only provide the barebone code to get you started. For more information, I strongly recommend going through the vignette and demo code.

```
## Split the dataset into 80:20 for training and validation
train_hex_split <- h2o.splitFrame(train_hex, ratios = 0.8, shuffle = TRUE)

## One Variable at at Time
ls_label <- c("Ca", "P", "pH", "SOC", "Sand")

for (n_label in 1:5) {

  ## Display
  cat("\n\nNow training a DNN model for", ls_label[n_label], "...\n")

  ## Train a 50-node, three-hidden-layer Deep Neural Networks for 100 epochs
  model <- h2o.deeplearning(x = 2:3595,
                            y = (3595 + n_label),
                            data = train_hex_split[[1]],
                            validation = train_hex_split[[2]],
                            activation = "Rectifier",
                            hidden = c(50, 50, 50),
                            epochs = 100,
                            classification = FALSE,
                            balance_classes = FALSE)

  ## Print the Model Summary
  print(model)

  ## Use the model for prediction and store the results in submission template
  raw_sub[, (n_label + 1)] <- as.matrix(h2o.predict(model, test_hex))

}
```

train/validation split

5 different regression targets

3594 features

Train Deep Learning model

check validation error

predict on test set

Step 2.5 - Save Results as a CSV

The final step is to save all predictions as a CSV file. This is the CSV file in the correct Kaggle submission format.

```
write.csv(raw_sub, file = "./results/my_Kaggle_submission.csv", row.names = FALSE)
```

save to CSV

$8,000 • 413 teams

Africa Soil Property Prediction Challenge

Enter/Merge by

Wed 27 Aug 2014 — Tue 21 Oct 2014 (40 days to go)

Dashboard

Leaderboard - Africa Soil Property Prediction Challenge

This leaderboard is calculated on approximately 13% of the test data.
The final results will be based on the other 87%, so the final standings may be different.

See someone using multiple accounts?
Let us know.

#	Δ3d	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑13	Jo-fai Chow @ blenditbayes! + h2o.ai + Domino *	0.40406	23	Thu, 11 Sep 2014 21:12:59 (-14.5h)
2	↑2	Pietro Marini *	0.40463	27	Thu, 11 Sep 2014 05:45:52 (-24.1h)
3	↑124	Lorenzo Rossi *	0.40548	11	Wed, 10 Sep 2014 16:12:41 (-26.8h)

Jo-fai made it to the very top with H2O Deep Learning!

(He probably used ensembles: stacking/blending, etc.)

My best ranking was 11th with a single H2O Deep Learning model (per target)

17	new	**Arno Candel H2O Deep Learning**	**0.42727**	**6**	**Thu, 11 Sep 2014 08:15:13 (-29.3h)**

BUT: Too early to tell: Leaderboard scores are computed on ~100 rows.

It's time for Live Modeling!

Challenges:

- no domain knowledge -> need to find/read papers
- small data -> need to apply regularization to avoid overfitting
- need to explore data and do feature engineering, if applicable

We will do the following steps:

- launch H2O from R
- import Kaggle data
- train separate Deep Learning model for 5 targets
- compute cross-validation error (N-fold)
- tune model parameters (grid search)
- do simple feature engineering (dimensionality reduction)
- build an ensemble model
- create a Kaggle submission

```r
library(h2o)

# Connect to H2O server (On server(s), run 'java -jar h2o.jar -Xmx4G -port 43322 -name AfricaSoil'
#h2oServer <- h2o.init(ip="mr-0xd1", port = 43322)

# Launch H2O directly on localhost
h2oServer <- h2o.init()

# Import data
path_train <- "/home/arno/kaggle_africasoil/data/training.csv.gz"
path_test <- "/home/arno/kaggle_africasoil/data/sorted_test.csv.gz"
path_output <- "/home/arno/kaggle_africasoil/outputs"
train_hex <- h2o.importFile(h2oServer, path = path_train)
test_hex <- h2o.importFile(h2oServer, path = path_test)

# Group variables
vars <- colnames(train_hex)
spectra <- vars[seq(2,3579,by=20)] # "poor man's dimensionality reduction": take every N-th column
extra <- vars[3580:3595]
targets <- vars[3596:3600]
predictors <- c(spectra, extra)

## Settings
ensemble_size <- 2
n_fold = 3

# LB score of 0.439
#ensemble_size <- 20
#n_fold = 20

# Scoring helpers
MSEs <- matrix(0, nrow = 1, ncol = length(targets))
RMSEs <- matrix(0, nrow = 1, ncol = length(targets))
CMRMSE = 0
```

```r
# Main loop over regression targets
for (resp in 1:length(targets)) {
  cat("\n\nNow training and cross-validating a DL model for", targets[resp], "...\n")

  # Run grid search with n-fold cross-validation
  cvmodel <-
    h2o.deeplearning(x = predictors,
                     y = targets[resp],
                     data = train_hex,
                     nfolds = n_fold,
                     classification = F,
                     activation="RectifierWithDropout",
                     hidden = c(100,100),
                     hidden_dropout_ratios = c(0.0,0.0),
                     input_dropout_ratio = 0,
                     epochs = 100,
                     l1 = c(0,1e-5),
                     l2 = c(0,1e-5),
                     rho = 0.99,
                     epsilon = 1e-8,
                     train_samples_per_iteration = -2
    )

  ## Collect cross-validation error
  MSE <- cvmodel@sumtable[[1]]$prediction_error   #If cvmodel is a grid search model
  #MSE <- cvmodel@model$valid_sqr_error            #If cvmodel is not a grid search model
  RMSE <- sqrt(MSE)
  CMRMSE <- CMRMSE + RMSE #column-mean-RMSE
  MSEs[resp] <- MSE
  RMSEs[resp] <- RMSE
  cat("\nCross-validated MSEs so far:", MSEs)
  cat("\nCross-validated RMSEs so far:", RMSEs)
  cat("\nCross-validated CMRMSE so far:", CMRMSE/resp)
```

```r
cat("\n\nTaking parameters from grid search winner for", targets[resp], "...\n")
p <- cvmodel@sumtable[[1]]  #If cvmodel is a grid search model
#p <- cvmodel@model$params   #If cvmodel is not a grid search model

## Build an ensemble model on full training data
for (n in 1:ensemble_size) {
  cat("\n\nBuilding ensemble model", n, "of", ensemble_size, "for", targets[resp], "...\n")
  model <-
    h2o.deeplearning(x = predictors,
                     y = targets[resp],
                     key = paste0(targets[resp], "_cv_ensemble_", n, "_of_", ensemble_size),
                     data = train_hex,
                     classification = F,
                     activation = p$activation,
                     hidden = p$hidden,
                     hidden_dropout_ratios = p$hidden_dropout_ratios,
                     input_dropout_ratio = p$input_dropout_ratio,
                     epochs = p$epochs,
                     l1 = p$l1,
                     l2 = p$l2,
                     rho = p$rho,
                     epsilon = p$epsilon,
                     train_samples_per_iteration = p$train_samples_per_iteration)

  ## Aggregate ensemble model predictions
  test_preds <- h2o.predict(model, test_hex)
  if (n == 1) {
    test_preds_blend <- test_preds
  } else {
    test_preds_blend <- cbind(test_preds_blend, test_preds[,1])
  }
}
```

```
  ## Now create submission
  cat (paste0("\n Number of ensemble models: ", ncol(test_preds_blend)))
  ensemble_average <- matrix("ensemble_average", nrow = nrow(test_preds_blend), ncol = 1)
  ensemble_average <- rowMeans(as.data.frame(test_preds_blend)) # Simple ensemble average
  ensemble_average <- as.data.frame(ensemble_average)

  colnames(ensemble_average)[1] <- targets[resp]
  if (resp == 1) {
    final_submission <- cbind(as.data.frame(test_hex[,1]), ensemble_average)
  } else {
    final_submission <- cbind(final_submission, ensemble_average)
  }
  print(head(final_submission))
}
cat(paste0("\nOverall cross-validated CMRMSE = " , CMRMSE/length(targets)))

## Write to CSV
write.csv(final_submission, file = "./submission.csv", quote = F, row.names=F)
```

https://github.com/0xdata/h2o/

0xdata / h2o

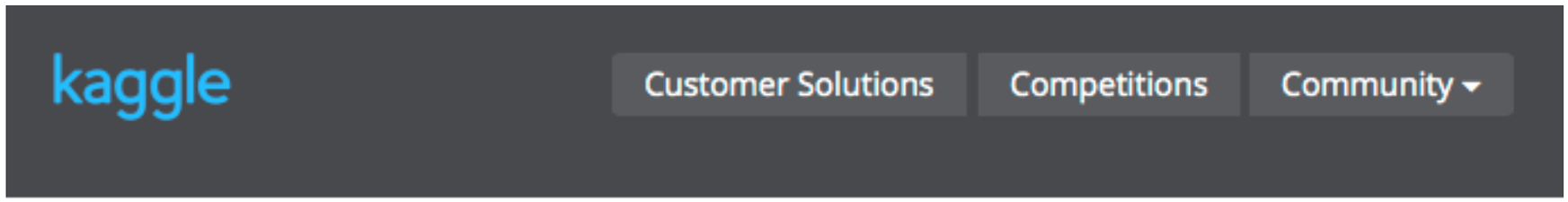
Unwatch 231

branch: master

h2o / R / examples / Kaggle / **MeetupKaggleAfricaSoil.R**

Live Modeling #2

Kaggle Display Advertising Challenge



Completed • $16,000 • 718 teams

Display Advertising Challenge

Tue 24 Jun 2014 – Tue 23 Sep 2014 (15 days ago)

Predict click-through rates on display ads

Display advertising is a billion dollar effort and one of the central uses of machine learning on the Internet. However, its data and methods are usually kept under lock and key. In this research competition, CriteoLabs is sharing a week's worth of data for you to develop models predicting ad click-through rate (CTR). Given a user and the page he is visiting, what is the probability that he will click on a given ad?

Real-world dirty data: Missing values, categorical values with >10k factor levels, etc.

kaggle — Customer Solutions | Competitions | Community

criteo labs

Completed • $16,000 • 718 teams

Display Advertising Challenge

Tue 24 Jun 2014 – Tue 23 Sep 2014 (15 days ago)

45.8M rows
41 columns
binary classifier

train.hex

41 columns, 45,840,617 rows, 2.29 GB bytes (compressed), 564.1 MB missing elements

Build models using Random Forest, BigData Random Forest, Distributed GBM, Generalized Linear Modeling, Deep Learning, Linear Regression, K-Means
Summary, Download as CSV, Export to file, Split frame, N-fold extract, ReBalance frame (load balancing)

0 Jump to row!

info |< < 0 1 2 3 4 5 6 7 8 9 10 11 > >|

Row	Id	Label	I1	I2	I3	I4	I5	I6	C17	C18	C19	C20
Change Type	As Factor	As Factor	As Factor	As Factor	As Factor	As Factor	As Factor	As Factor				
Type	Int	Int	Int	Int	Int	Int	Int	Int	Enum	Enum	Enum	Enum
Min	10000000	0	0	-3	0	0	0	0				
Max	55840616	1	5775	257675	65535	969	23159456	431037				
Mean	32920308	0.256	3.502	105.848	26.913	7.323	18538.992	116.062				
Std Dev	13233046.427	0.437	9.429	391.458	397.973	8.793	69394.602	382.566				
Cardinality									10	5652	2172	3
Missing			20793556		9839447	9937369	1183117	10252328			20172858	20172858
0	10000000	0	1	1	5	0	1382	4	e5ba7672	f54016b9	21ddcdc9	b1252a9d
1	10000001	0	2	0	44	1	102	8	07c540c4	b04e4670	21ddcdc9	5840adea
2	10000002	0	2	0	1	14	767	89	8efede7f	3412118d	-	-
3	10000003	0	-	893	-	-	4392	-	1e88c74f	74ef3502	-	-
4	10000004	0	3	-1	-	0	2	0	1e88c74f	26b3c7a7	-	-
5	10000005	0	-	-1	-	-	12824	-	776ce399	92555263	-	-
6	10000006	0	-	1	2	-	3168	-	776ce399	cdfa8259	-	-
7	10000007	1	1	4	2	0	0	0	e5ba7672	74ef3502	-	-
8	10000008	0	-	44	4	8	19010	249	e5ba7672	42a2edb9	-	-
9	10000009	0	-	35	-	1	33737	21	d4bb7bd8	70d0f5f9	-	-
10	10000010	0	-	2	632	0	56770	-	776ce399	0b331314	21ddcdc9	5840adea

H2O Deep Learning expands categorical factor levels (on the fly, but still):
41 features turn into N=47,411 input values for neural network

Progress

Status of Neuron Layers

#	Units	Type	Dropout	L1	L2	Rate (Mean, RMS)	Weight (Mean, RMS)	Bias (Mean, RMS)
1	47411	Input	0.00 %					
2	200	Rectifier	0.00 %	0.0	0.0	(0.825013, 0.377112)	(-7.99101e-06, 0.00648171)	(0.0758910, 0.119361)
3	200	Rectifier	0.00 %	0.0	0.0	(0.0294500, 0.0741626)	(-0.00221767, 0.0698427)	(0.991344, 0.0170028)
4	2	Softmax		0.0	0.0	(0.00189123, 0.00210229)	(0.0253094, 0.390903)	(7.71869e-05, 0.00426712)

Classification error on training data: 37.10 %

Classification error on validation data: 36.57 %

Training samples: 50,401

Epochs: 0.001 / 10.000

Number of compute nodes: 10 (320 threads)

Training samples per iteration (user-given): 10,000

Training speed: 436 samples/s

Training time: 1 min 55.358 sec

fully categorical expansion:
training is slow (also more communication)

Use hashing trick to reduce dimensionality N:

naive: input[key_to_idx(key)] = 1 //one-hot encoding

hash: input[hash_fun(key) % M] += 1, where M << N

E.g, M=1000 —> much faster training, rapid convergence!

Progress

Status of Neuron Layers

#	Units	Type	Dropout	L1	L2	Rate (Mean, RMS)	Weight (Mean, RMS)	Bias (Mean, RMS)
1	1000	Input	0.00 %					
2	200	Rectifier	0.00 %	0.0	0.0	(0.00539496, 0.00215323)	(-0.00152944, 0.0422612)	(0.0316117, 0.0693198)
3	200	Rectifier	0.00 %	0.0	0.0	(0.158038, 0.190765)	(-0.0182553, 0.0729768)	(0.119384, 0.249879)
4	2	Softmax		0.0	0.0	(0.0125651, 0.00803911)	(0.0250770, 0.298090)	(-0.00718224, 0.0108469)

Classification error on training data: 30.59 %

Classification error on validation data: 33.44 %

Training samples: 5,999,521

Epochs: 0.132 / 10.000

Number of compute nodes: 10 (320 threads)

Training samples per iteration (user-given): 1,000,000

Training speed: 221,466 samples/s

Training time: 27.090 sec

Note:

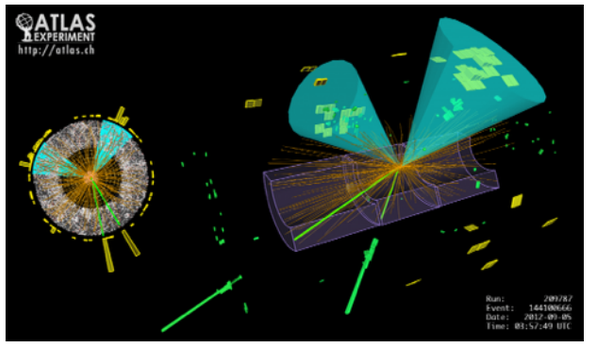
Hashing trick is also used by winners of Kaggle challenge, and the 4-th ranked submission uses an **Ensemble of Deep Learning models** with some feature engineering (drop rare categories etc)

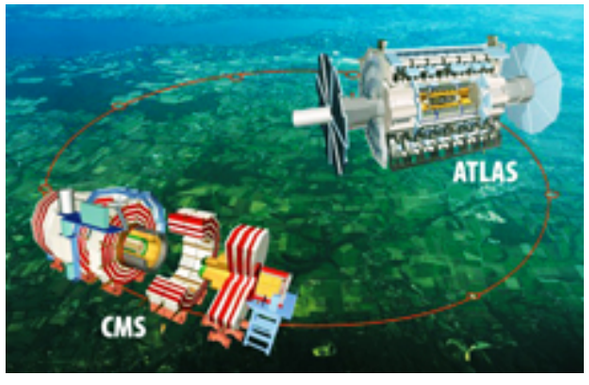
Live Modeling #3

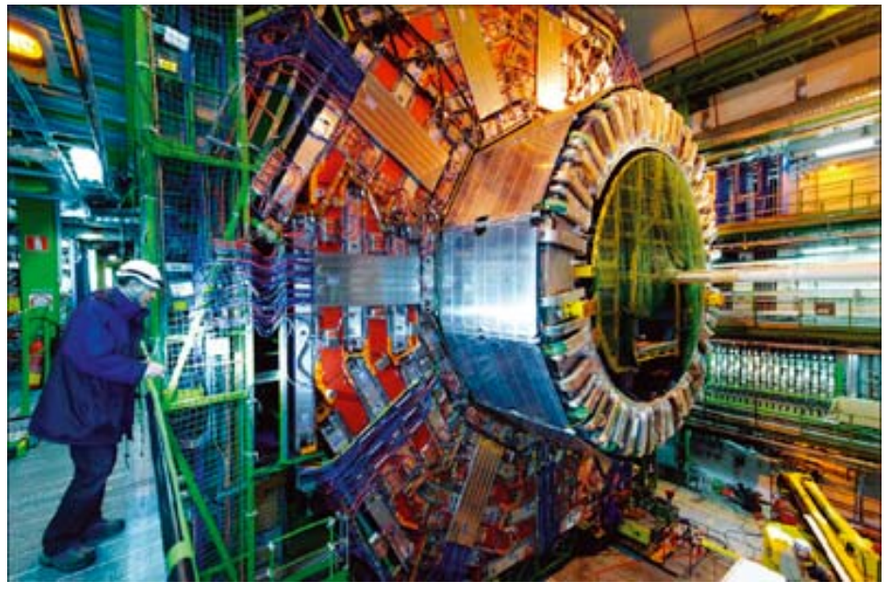
Kaggle Higgs Boson Machine Learning Challenge

Higgs Particle Discovery

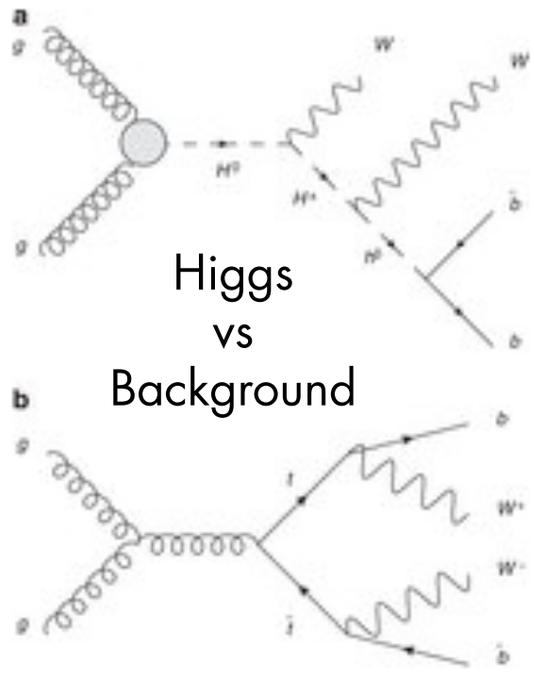
Large Hadron Collider: Largest experiment of mankind!
$13+ billion, 16.8 miles long, 120 MegaWatts, -456F, 1PB/day, etc.
Higgs boson discovery (July '12) led to 2013 Nobel prize!







Images courtesy CERN / LHC



I missed the deadline for Kaggle by 1 day! But still want to get good numbers!

Higgs challenge

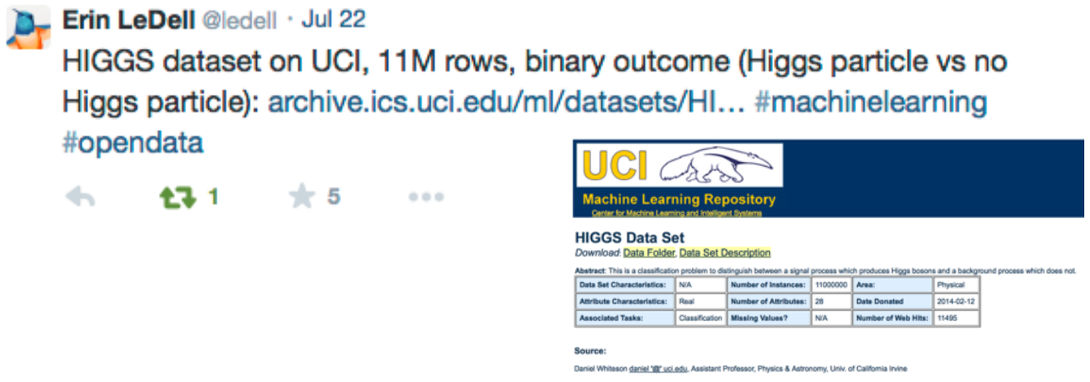
Completed • $13,000 • 1,785 teams

Higgs Boson Machine Learning Challenge

Mon 12 May 2014 – Mon 15 Sep 2014 (23 days ago)

Higgs: Binary Classification Problem

<u>HIGGS UCI Dataset:</u>
21 low-level features AND
7 high-level derived features
Train: 10M rows, Test: 500k rows



Conventional methods of choice for physicists:
- Boosted Decision Trees
- Neural networks with 1 hidden layer

BUT: Must first add derived high-level features (physics formulae)

Metric: AUC = Area under the ROC curve (range: 0.5…1, higher is better)

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

Higgs: Can Deep Learning Do Better?

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
Deep Learning	**?**	**?**

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

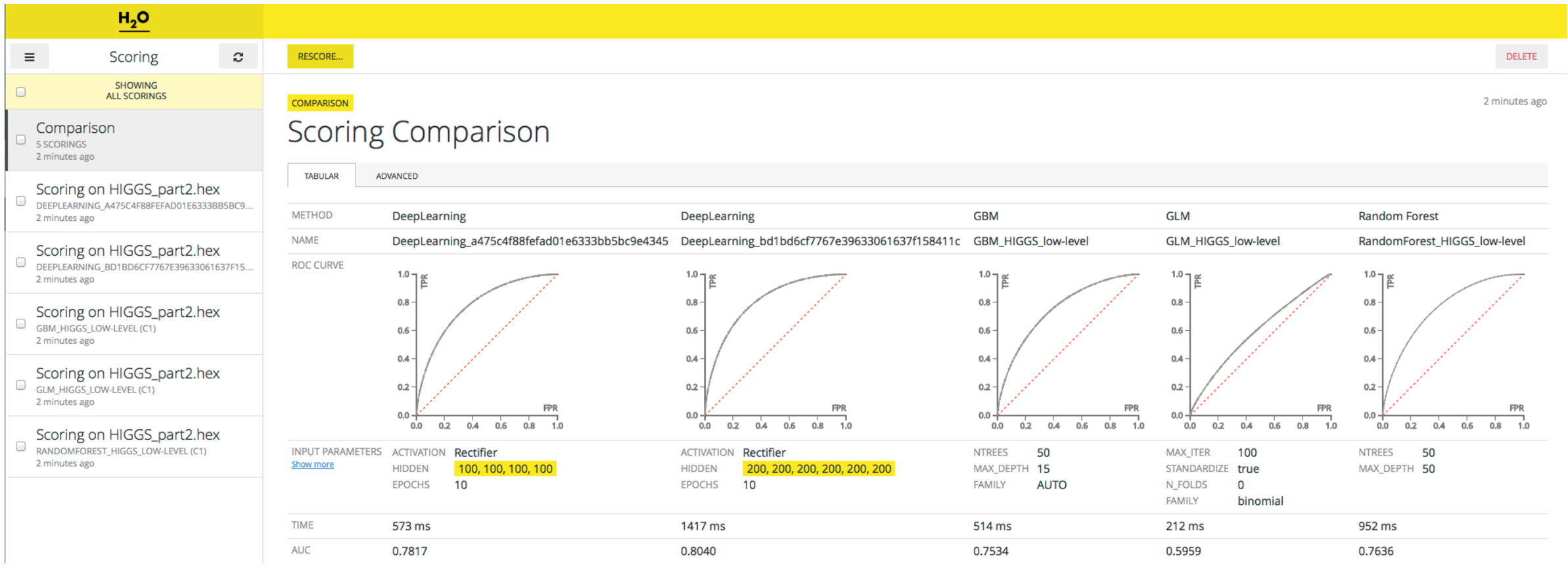
	AUC		
Technique	Low-level	High-level	Complete
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	<Your guess goes here>		

reference paper results: baseline 0.733

Let's build a H_2O Deep Learning model and find out!

H_2O Steam: Scoring Platform

http://server:port/steam/index.html

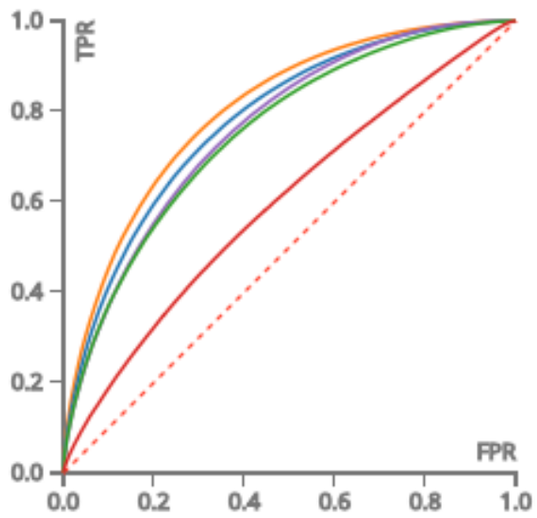


Higgs Dataset Demo on 10-node cluster
Let's score all our H_2O models and compare them!

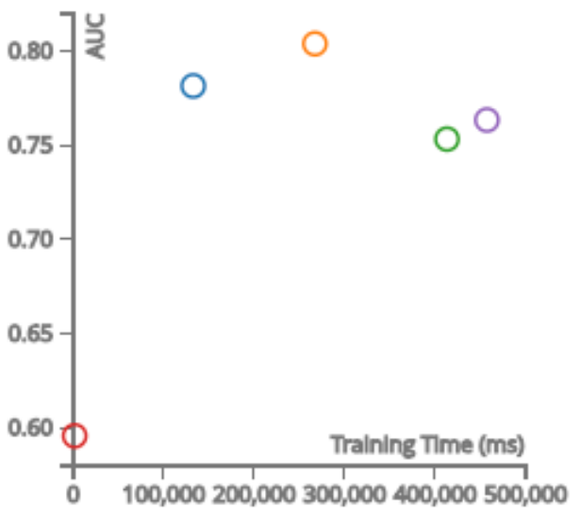
Scoring Higgs Models in H_2O Steam

Visualizations

ROC CHART



TRAINING TIME (MS) VS AUC

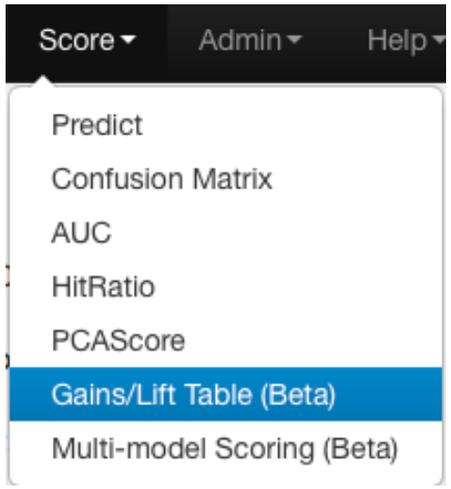


Comparison

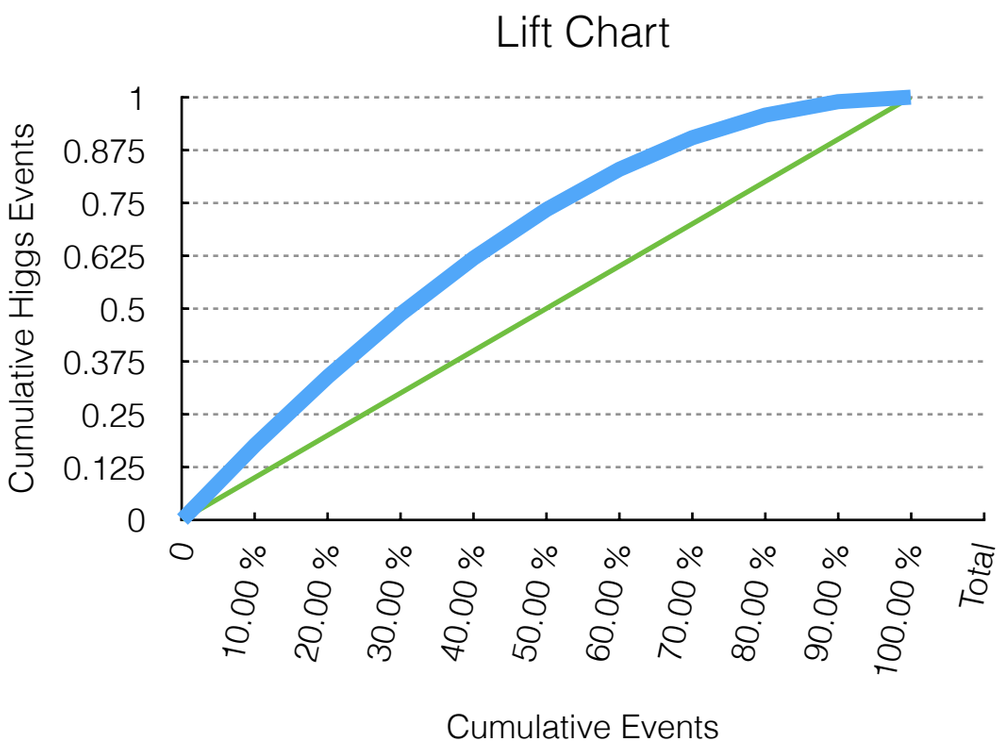
METHOD	AUC	GINI	TRAINING TIME (MS)	SCORING TIME (MS)
DeepLearning	0.8040	0.6081	266936	1417
DeepLearning	0.7817	0.5633	132442	573
Random Forest	0.7636	0.5271	457052	952
GBM	0.7534	0.5069	412887	514
GLM	0.5959	0.1918	1407	212

Live Demo on 10-node cluster:
<10 minutes runtime for all algos!
Better than LHC baseline of AUC=0.73!

Lift chart demo for simple Higgs model



Quantile	Response rate	Lift	Cumulative lift
10.00%	93.55%	1.7683085	17.68%
20.00%	85.76%	1.6210535	33.89%
30.00%	78.39%	1.4817379	48.71%
40.00%	69.86%	1.3206078	61.92%
50.00%	60.80%	1.1492702	73.41%
60.00%	50.38%	0.9523756	82.93%
70.00%	39.52%	0.7470124	90.40%
80.00%	28.24%	0.5337477	95.74%
90.00%	16.88%	0.319122	98.93%
100.00%	5.65%	0.10676467	100.00%
Total	52.90%	1.0	



Progress

Status of Neuron Layers

#	Units	Type	Dropout
1	21	Input	0.00 %
2	200	Rectifier	0.00 %
3	200	Rectifier	0.00 %
4	200	Rectifier	0.00 %
5	200	Rectifier	0.00 %
6	200	Rectifier	0.00 %
7	200	Rectifier	0.00 %
8	2	Softmax	

Classification error on training data: 24.51 %

Classification error on validation data: 26.62 %

Training samples: 502,328,135

Epochs: 50.233 / 110.001

Number of compute nodes: 10 (320 threads)

Training samples per iteration (-2 -> auto-tuning): 5,417,745

Training speed: 297,829 samples/s

Training time: 28 min 19.108 sec

Estimated time left: 33 min 41.623 sec

Higgs Particle Detection with H_2O

HIGGS UCI Dataset:
21 **low-level** features AND
7 **high-level** derived features
Train: 10M rows, Test: 500k rows

	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)

nature http://arxiv.org/pdf/1402.4735v2.pdf

Algorithm	Paper's l-l AUC	**low-level** H2O AUC	all features H2O AUC	Parameters (not heavily tuned), H2O running on 10 nodes
Generalized Linear Model	–	0.596	0.684	default, binomial
Random Forest	–	0.764	0.840	50 trees, max depth 50
Gradient Boosted Trees	0.73	0.753	0.839	50 trees, max depth 15
Neural Net 1 layer	0.733	0.760	0.830	1x300 Rectifier, 100 epochs
Deep Learning 3 hidden layers	0.836	**0.850**	–	3x1000 Rectifier, L2=1e-5, 40 epochs
Deep Learning 4 hidden layers	0.868	**0.869**	–	4x500 Rectifier, L1=L2=1e-5, 300 epochs
Deep Learning 6 hidden layers	0.880	running	–	6x500 Rectifier, L1=L2=1e-5

Deep Learning on low-level features alone beats everything else!
H_2O prelim. results compare well with paper's results* (TMVA & Theano)

What methods led the Higgs Boson Kaggle contest?

Winning 3 submissions:

1. Bag of 70 **dropout** (50%) **deep neural networks** (600,600,600), **channel-out** activation function, l1=5e-6 and l2=5e-5 (for first weight matrix only). **Each input neurons is connected to only 10 input values** (sampled with replacement before training).
2. A **blend** of boosted decision tree ensembles constructed using Regularized Greedy Forest
3. **Ensemble** of 108 rather small **deep neural networks** (30,50), (30,50,25), (30,50,50,25)

Ensemble Deep Learning models are winning!
Check out h2o/R/ensemble by Erin LeDell!

H2O.ai

Deep Learning Tips & Tricks

General:
More layers for more complex functions (exp. more non-linearity).
More neurons per layer to detect finer structure in data ("memorizing").
Validate model performance on holdout data.
Add some regularization for less overfitting (lower validation set error).
Ensembles typically perform better than individual models.

Specifically:
Do a grid search to get a feel for convergence, then continue training.
Try Tanh/Rectifier, try max_w2=10…50, L1=1e-5..1e-3 and/or L2=1e-5…1e-3
Try Dropout (input: up to 20%, hidden: up to 50%) with test/validation set. Input dropout is recommended for noisy high-dimensional input.
Distributed: More training samples per iteration: faster, but less accuracy?
With ADADELTA: Try epsilon = 1e-4,1e-6,1e-8,1e-10, rho = 0.9,0.95,0.99
Without ADADELTA: Try rate = 1e-4…1e-2, rate_annealing = 1e-5…1e-9,
momentum_start = 0.5…0.9, momentum_stable = 0.99,
momentum_ramp = 1/rate_annealing.
Try balance_classes = true for datasets with large class imbalance.
Enable force_load_balance for small datasets.
Enable replicate_training_data if each node can hold all the data.

H2O.ai

You can participate!

- Image Recognition: Convolutional & Pooling Layers PUB-644
- Faster Training: GPGPU support PUB-1013
- NLP: Recurrent Neural Networks PUB-1052
- Unsupervised Pre-Training: Stacked Auto-Encoders PUB-1014
- Benchmark vs other Deep Learning packages
- Investigate other optimization algorithms
- Win Kaggle competitions with H2O!

More info

http://h2o.ai
http://0xdata.com/blog/
http://bit.ly/deeplearningh2o
http://www.slideshare.net/0xdata/
https://www.youtube.com/user/0xdata
https://github.com/0xdata/h2o
https://github.com/0xdata/h2o-dev
@hexadata
@ArnoCandel

Join us at H2O World 2014!!!