Deep Learning Through Examples

Arno Candel, H2O.ai

Silicon Valley Big Data Science Meetup
Mountain View, 2/12/2015

Who am I?

@ArnoCandel

PhD in Computational Physics, 2005
from ETH Zurich Switzerland

6 years at SLAC - Accelerator Physics Modeling
2 years at Skytree - Machine Learning
13 months at H2O.ai - Machine Learning

15 years in Supercomputing & Modeling

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

KDnuggets™

search KDnuggets | Search

advanced search | Help

Latest News

- → KDnuggets™ News 15:n03, Jan 28
- → KDnuggets Free Pass to Strata Hadoop World San Jose, Feb 17-20, 2015
- → Interview: Nandu Jayakumar, Yahoo on What Does One Need for Big Data Success
- → PAW: Predictive Analytics World and Text Analytics World, Spring 2015, San Francisco
- → GoodData Insights as a Service guides users thru the analytics process

Top stories on KDnuggets

g+1 2 | Share 7 | Tweet 3

Most viewed, Jan 18-24

1. Interview: Arno Candel, H2O.ai on the Basics of Deep Learning to Get You Started
2. Top SlideShare Presentations on Big Data, updated - Jan 18, 2015.
3. Top KDnuggets tweets, Jan 19-20: 15 programming languages you need to know in 2015; R Programming fun: writing a Twitter bot - Jan 21, 2015.
4. Text Analysis 101: Document Classification - Jan 24, 2015.
5. Top /r/MachineLearning posts, Jan 11-17 - Jan 18, 2015.
6. How to interview a data scientist - Jan 16, 2015.
7. Interview: Arno Candel, H20.ai on How to Quick Start Deep Learning with H2O - Jan 21, 2015.

Most shared, Jan 18-24

1. Interview: Arno Candel, H2O.ai on the Basics of Deep Learning to Get You Started - Jan 20, 2015.
2. Top SlideShare Presentations on Big Data, updated
3. 8 Trends In Big Data For 2015 - Jan 21, 2015.
4. Genetics as a Social Network - A Data Scientist Perspective - Jan 19, 2015.
5. Interview: Arno Candel, H20.ai on How to Quick Start Deep Learning with H2O - Jan 21, 2015.
6. Microsoft buys Revolution Analytics - Jan 24, 2015.
7. Interview: Arno Candel, H2O.ai on the Journey from Physics to Machine Learning - Jan 22, 2015.

Previous weeks top stories:

- Jan 11-17: Research Leaders on Data Mining key trends, top papers; Deep Learning in a Nutshell
- Jan 4-10: 11 Clever Methods of Overfitting; Research Leaders on Data Science and Big Data
- Dec 28 - Jan 3: What will happen to big data, data science? Analytics: 5 Rules to cut thru the Hype

Outline

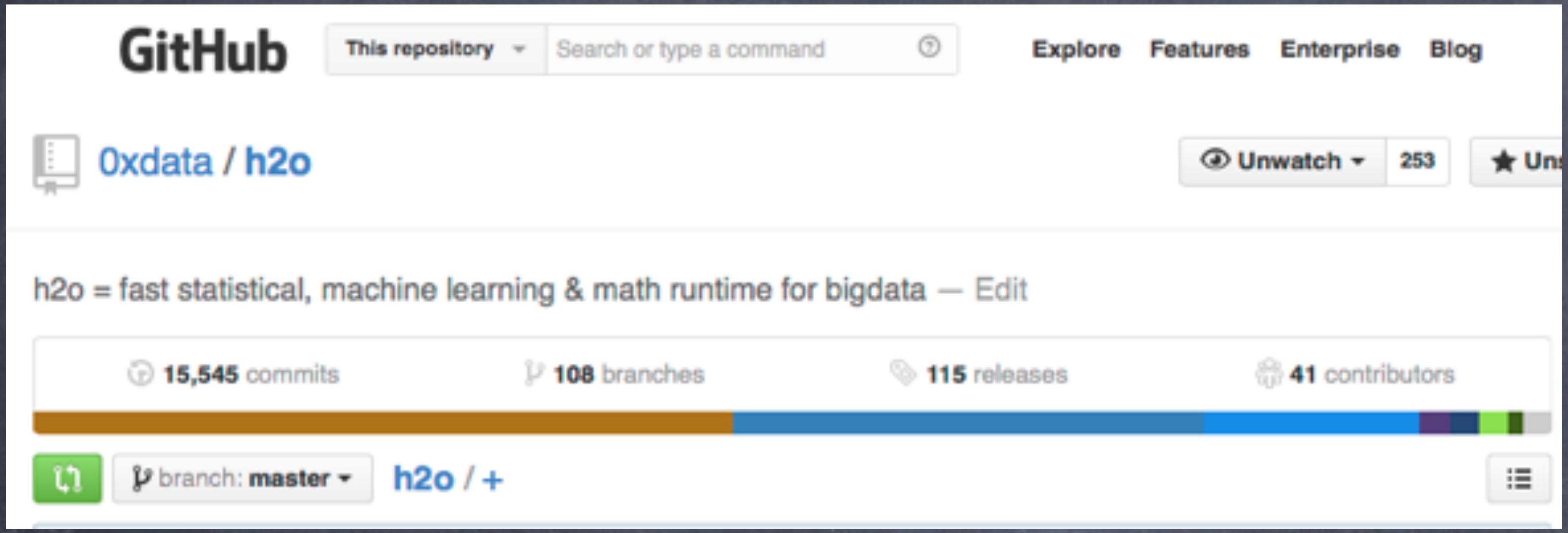
- Introduction (10 mins)
- Methods & Implementation (20 mins)
- Results and Live Demos (20 mins)
    - Higgs boson classification
    - MNIST handwritten digits
    - Ebay text classification
    - h2o-dev Outlook: Flow, Python
- Part 2: Hands-On Session (40 mins)
    - Web GUI: Higgs dataset
    - R Studio: Adult, Higgs, MNIST datasets

Teamwork at H2O.ai

Java, Apache v2 Open-Source

#1 Java Machine Learning in Github

Join the community!



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

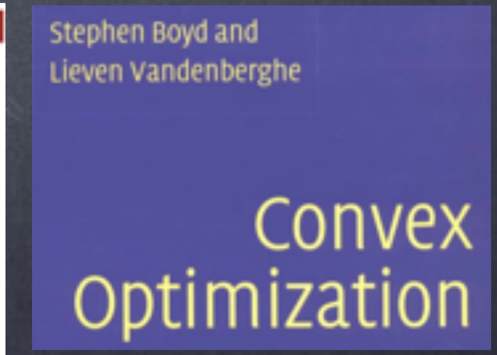
Investors

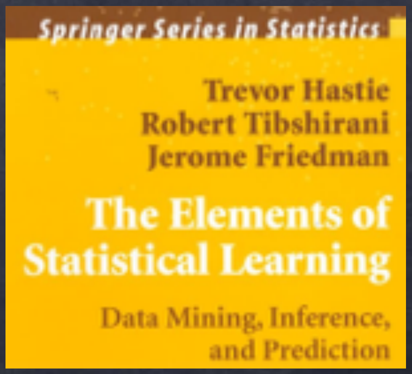
Jishnu Bhattacharjee
Nexus Venture Partners

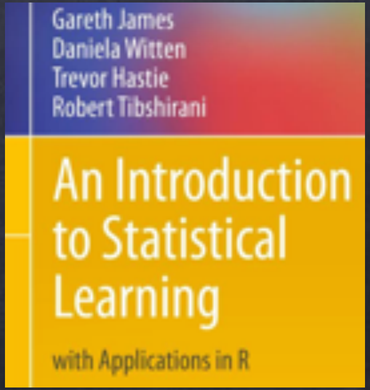
Anand Babu Periasamy
Founder, Gluster (RedHat)

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

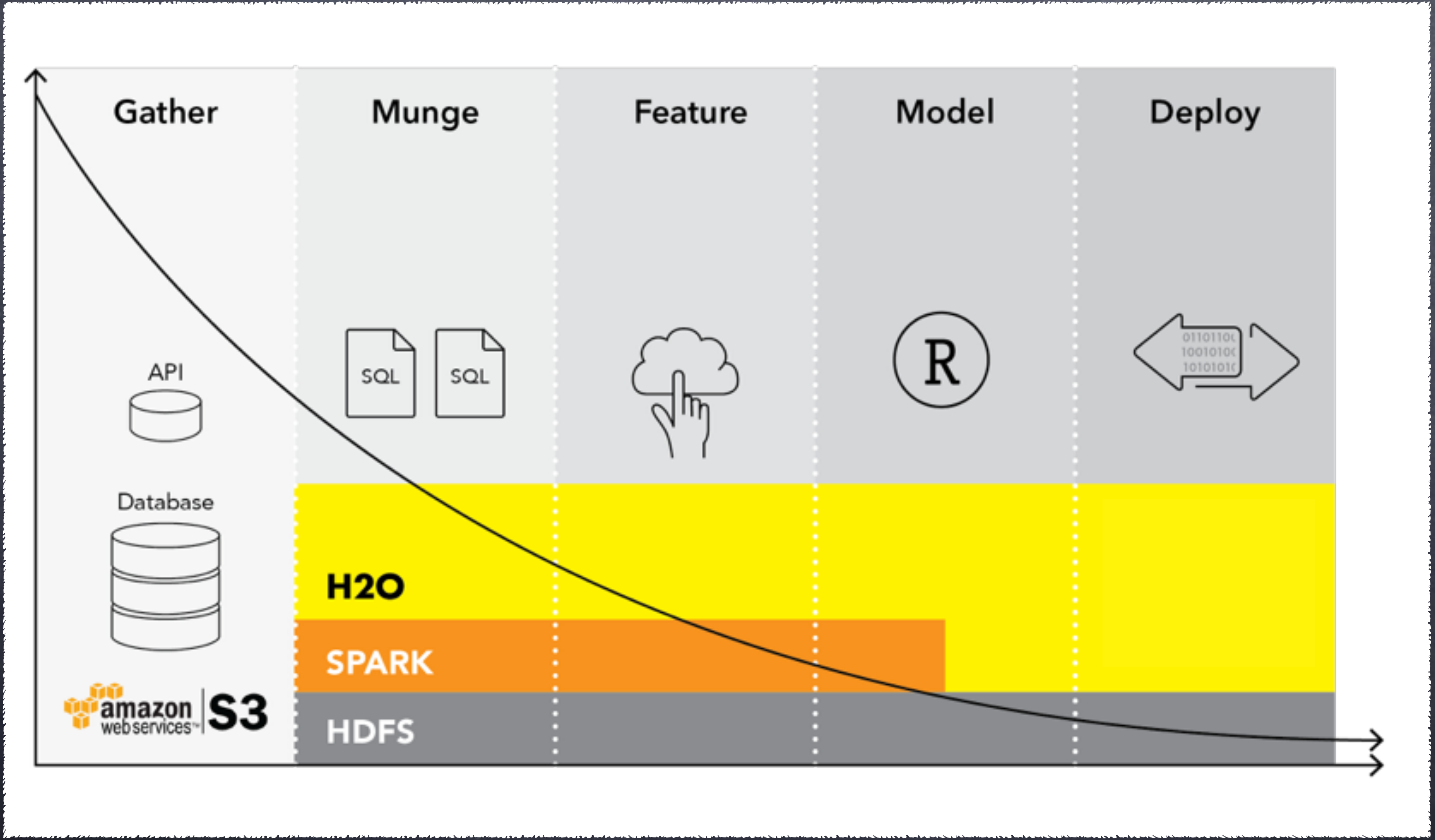
Dipchand "Deep" Nishar
SVP of Products & UX (LinkedIn)



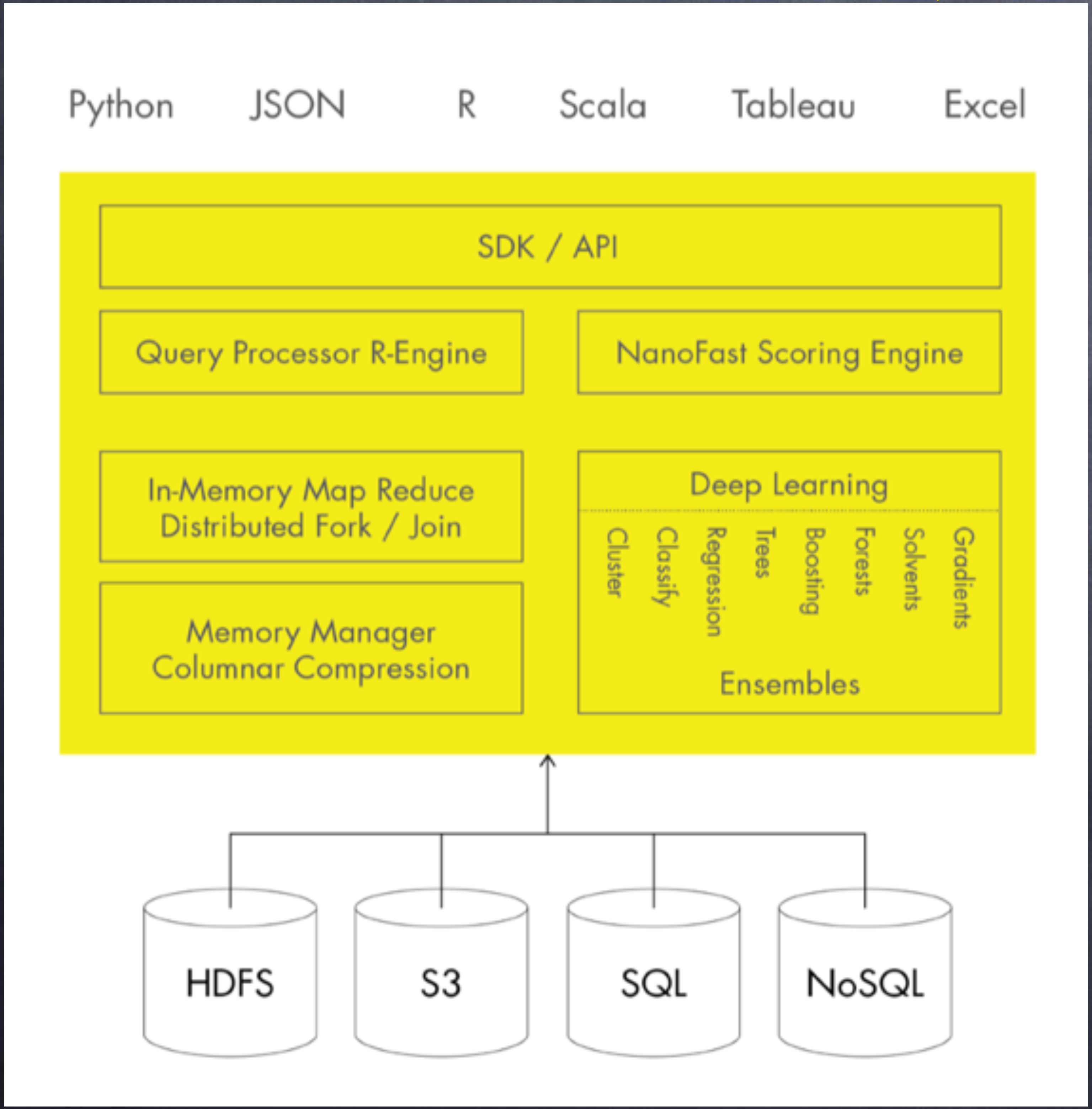




H2O: Open-Source (Apache v2) Predictive Analytics Platform



H2O Architecture - Designed for speed, scale, accuracy & ease of use



Key technical points:

- distributed JVMs + REST API
- no Java GC issues (data in byte[], ~~Double~~)
- loss-less number compression
- Hadoop integration (v1,YARN)
- R package (CRAN)

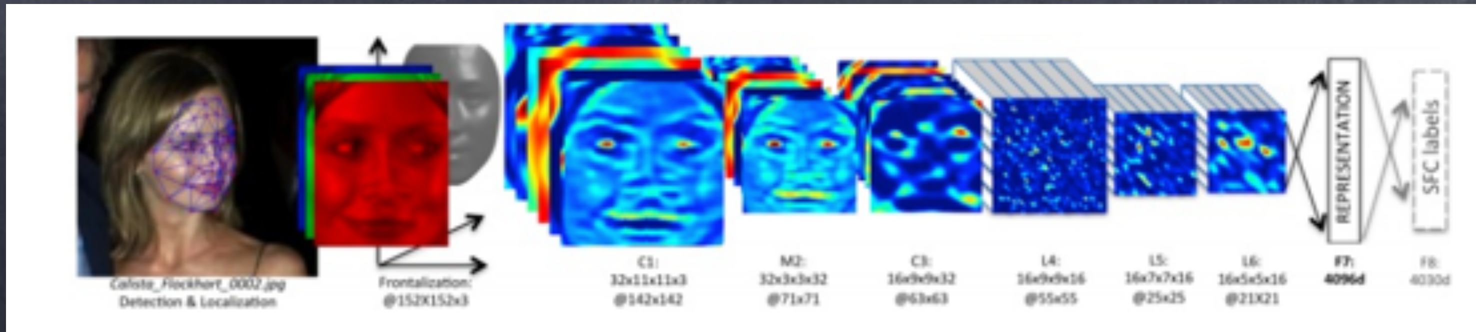
Pre-built fully featured algos: K-Means, NB, PCA, CoxPH, GLM, RF, GBM, DeepLearning

What is Deep Learning?

Wikipedia:

Deep learning is a set of algorithms in machine learning that attempt to model high-level abstractions in data by using architectures composed of multiple non-linear transformations.

Input: Image



DeepFace rotating a celebrity's face to use for facial verification. Image credit: Facebook

Output: User ID

Example: Facebook DeepFace

What is NOT Deep

Linear models are not deep
(by definition)

Neural nets with 1 hidden layer are not deep
(only 1 layer - no feature hierarchy)

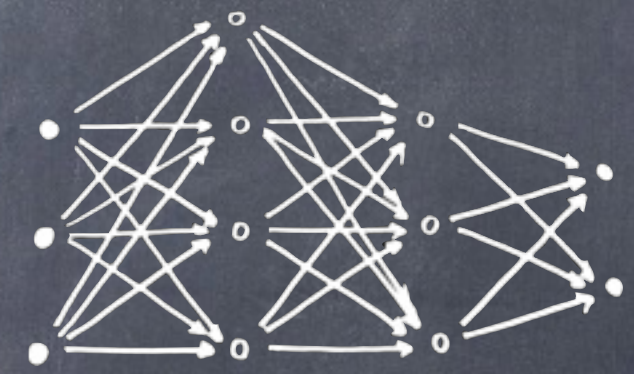
SVMs and Kernel methods are not deep
(2 layers: kernel + linear)

Classification trees are not deep

(operate on original input space, no new features generated)

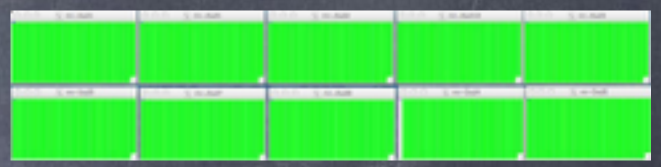
H2O Deep Learning

1970s multi-layer feed-forward Neural Network
(stochastic gradient descent with back-propagation)



+ 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)



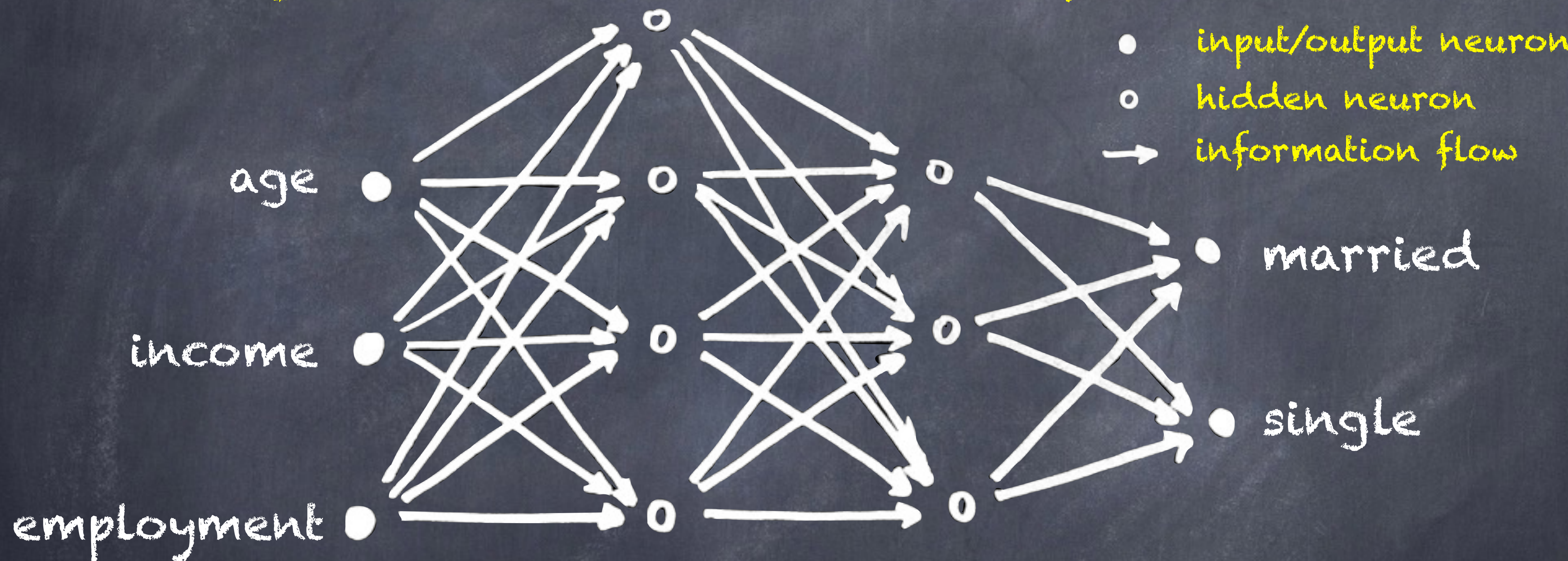
all 320 cores maxed out

+ 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

Example Neural Network

"fully connected" directed graph of neurons

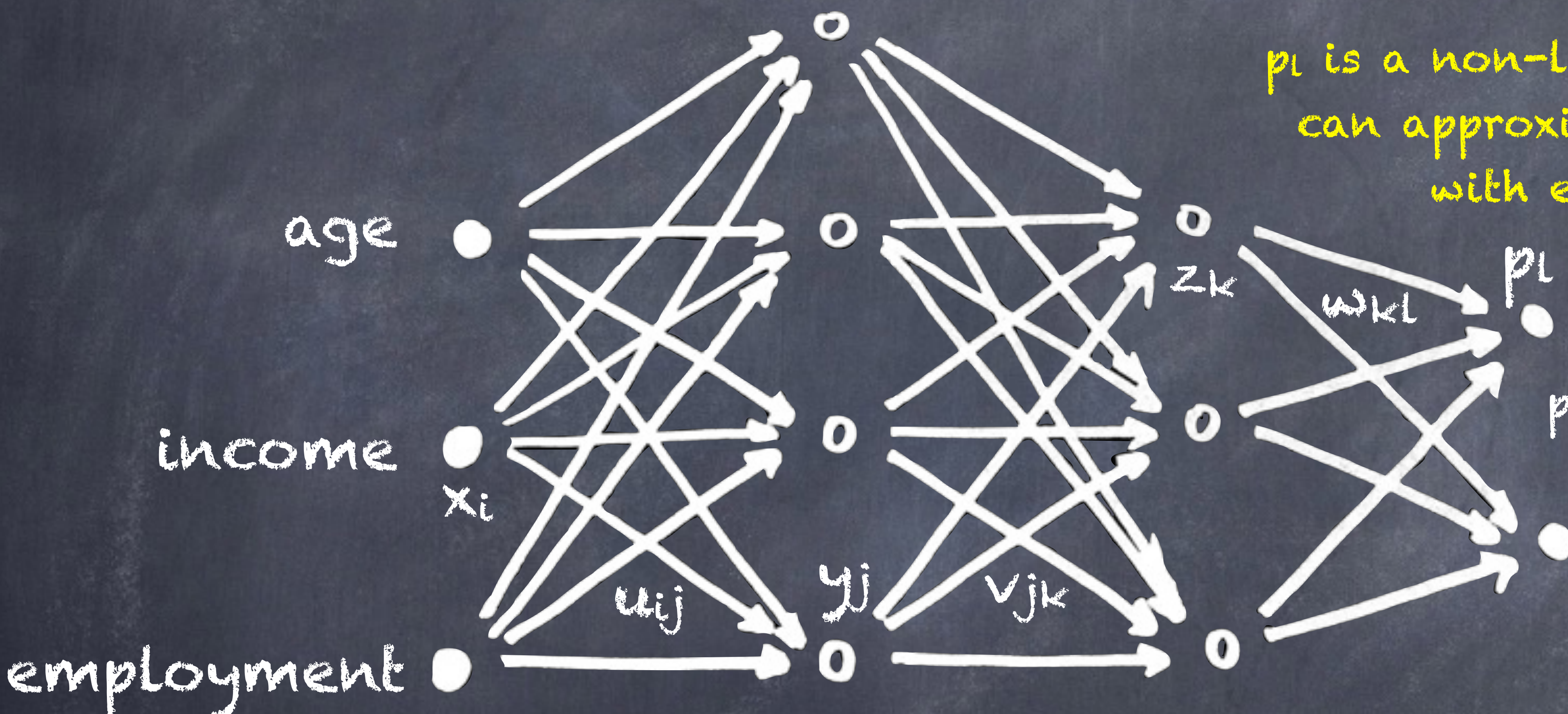


	Input layer	Hidden layer 1	Hidden layer 2	Output layer
#neurons	3	4	3	2
#connections	3x4	4x3	3x2	

Prediction: Forward Propagation

"neurons activate each other via weighted sums"

p_l is a non-linear function of x_i: can approximate ANY function with enough layers!



married

per-class probabilities
$\text{sum}(p_l) = 1$

single

$y_j = \tanh(\text{sum}_i(x_i * u_{ij}) + b_j)$

b_j, c_k, d_l: bias values (indep. of inputs)

$z_k = \tanh(\text{sum}_j(y_j * v_{jk}) + c_k)$

activation function: tanh

$p_l = \text{softmax}(\text{sum}_k(z_k * w_{kl}) + d_l)$

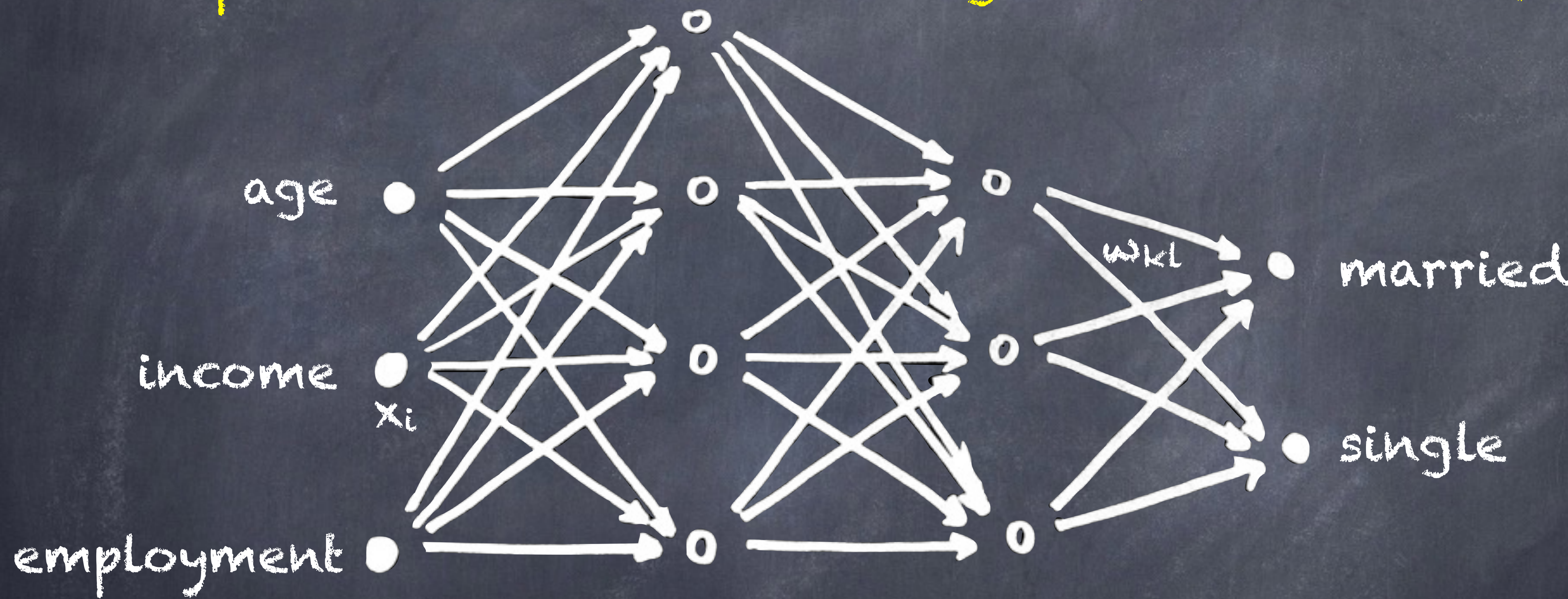
alternative:

x -> max(0,x) "rectifier"

$\text{softmax}(x_k) = \exp(x_k) / \text{sum}_k(\exp(x_k))$

Data preparation & Initialization

Neural Networks are sensitive to numerical noise, operate best in the linear regime (not saturated)



Automatic standardization of data

x_i: mean = 0, stddev = 1

horizontalize categorical variables, e.g.
{full-time, part-time, none, self-employed}
->
{0,1,0} = part-time, {0,0,0} = self-employed

Automatic initialization of weights

Poor man's initialization: random weights w_{kl}

Default (better): Uniform distribution in
+/- sqrt(6/(#units + #units_previous_layer))

Training: Update Weights & Biases

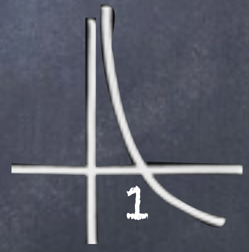
For each training row, we make a prediction and compare with the actual label (supervised learning):

predicted	actual	
0.8	1	married
0.2	0	single

Objective: minimize prediction error (MSE or cross-entropy)

Mean Square Error = $(0.2^2 + 0.2^2)/2$ "penalize differences per-class"

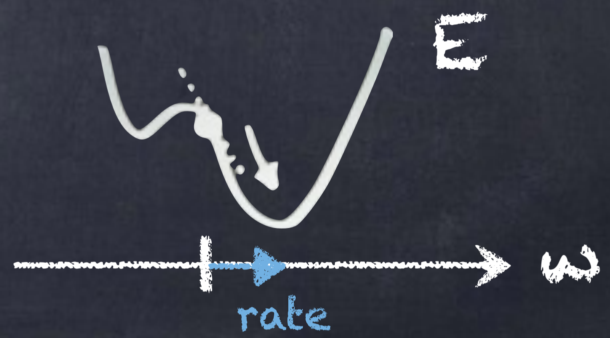
Cross-entropy = $-\log(0.8)$



"strongly penalize non-1-ness"

Stochastic Gradient Descent: Update weights and biases via gradient of the error (via back-propagation):

$$w \leftarrow w - \text{rate} * \partial E/\partial w$$

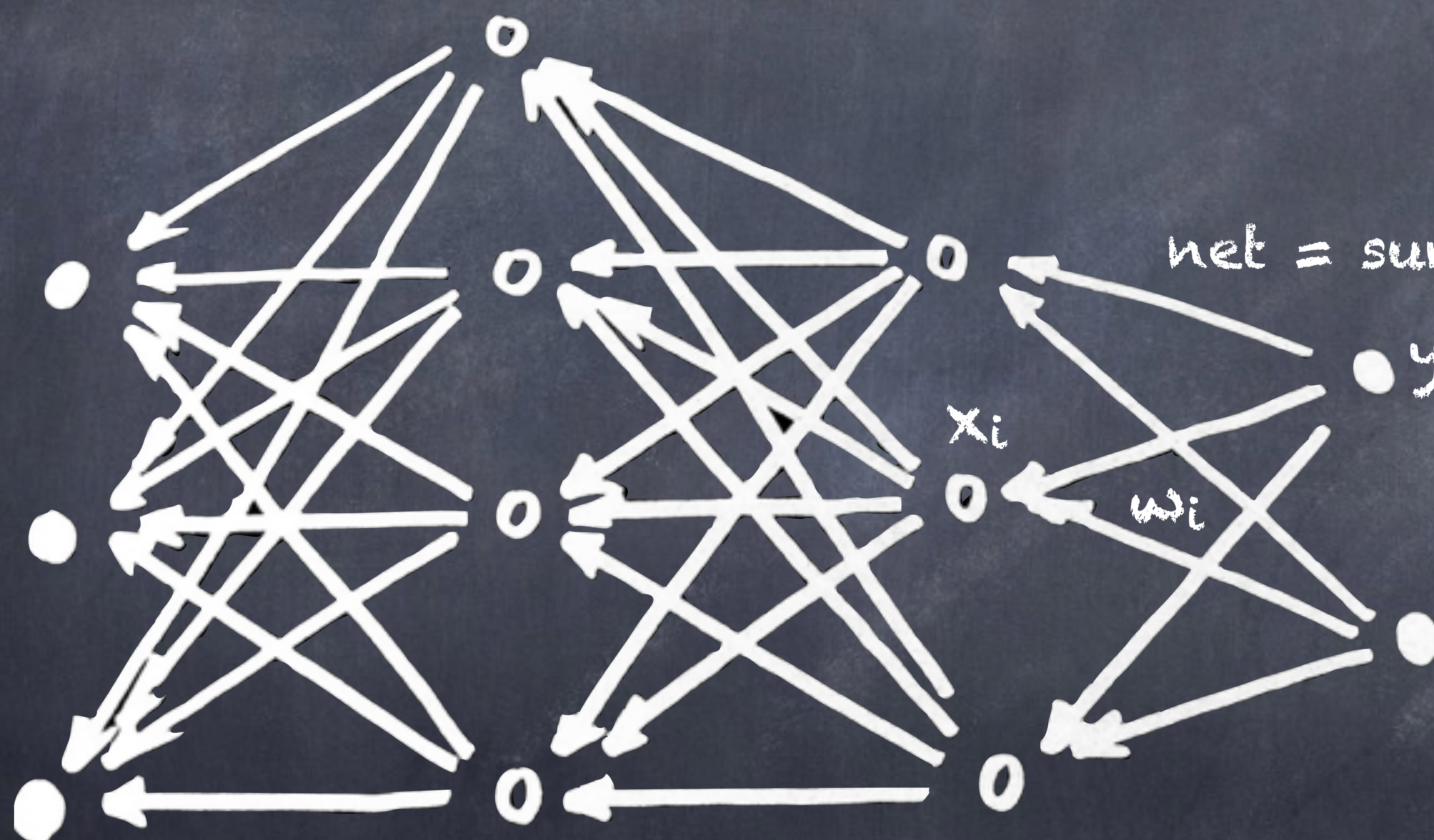


Backward Propagation

How to compute $\partial E/\partial w_i$ for $w_i \leftarrow w_i - rate * \partial E/\partial w_i$?

Naive: For every i, evaluate E twice at $(w_1,\ldots,w_i \pm \Delta,\ldots,w_N)$... Slow!

Backprop: Compute $\partial E/\partial w_i$ via chain rule going backwards



net = sum_i(wi*xi) + b

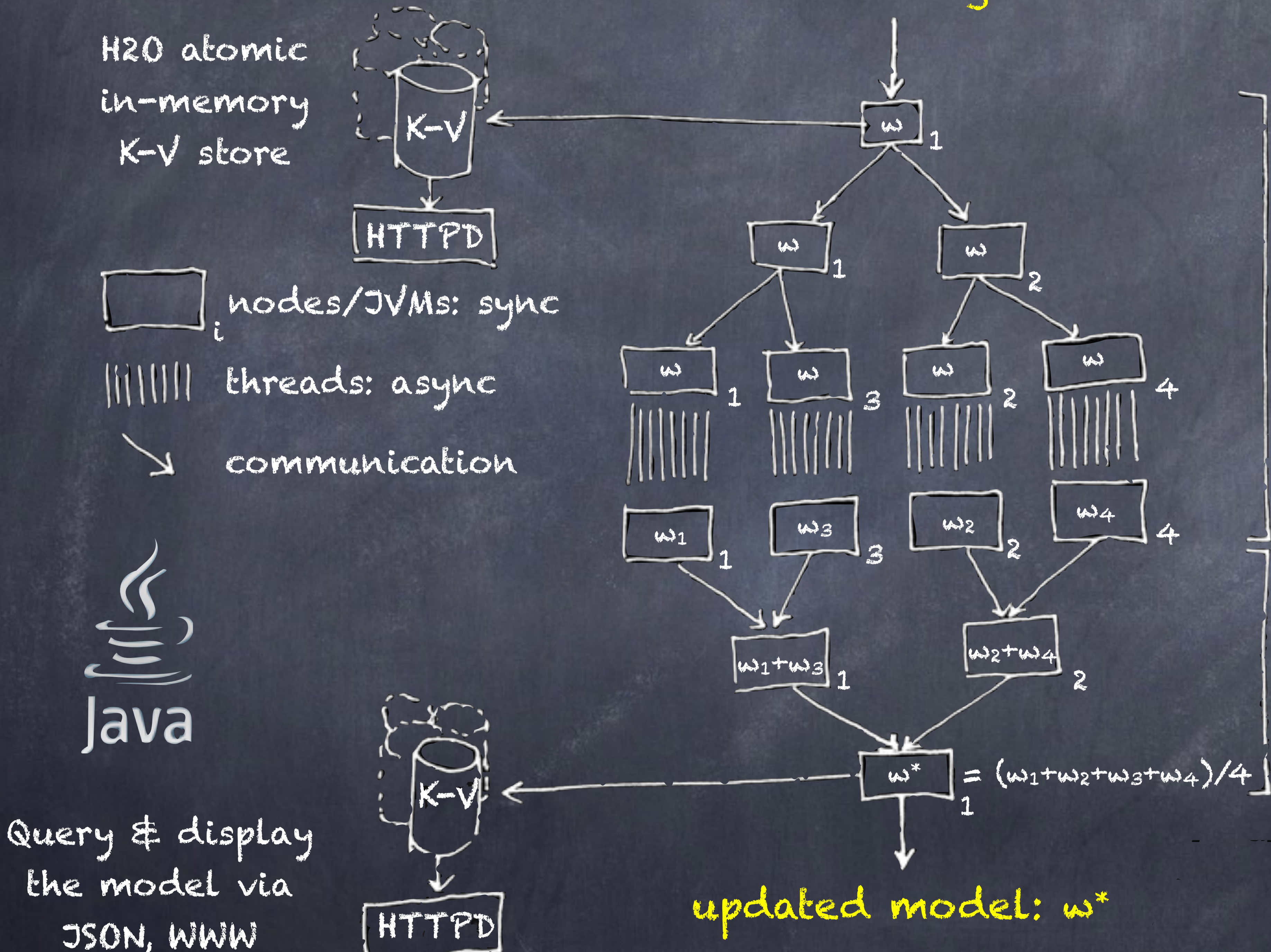
y = activation(net)

E = error(y)

$$\partial E/\partial w_i = \partial E/\partial y * \partial y/\partial net * \partial net/\partial w_i$$
$$= \partial(error(y))/\partial y * \partial(activation(net))/\partial net * xi$$

H2O Deep Learning Architecture

initial model: weights and biases w



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

reduce:
model averaging: average weights and biases from all nodes, speedup is at least #nodes/log(#rows)
arxiv:1209.4129v3

Keep iterating over the data ("epochs"), score from time to time

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

"Secret" Sauce to Higher Accuracy

Adaptive learning rate - ADADELTA (Google)

Automatically set learning rate for each neuron based on its training history

Regularization

L1: penalizes non-zero weights
L2: penalizes large weights
Dropout: randomly ignore certain inputs
Hogwild!: intentional race conditions
Distributed mode: weight averaging

Grid Search and Checkpointing

Run a grid search to scan many hyper-parameters, then continue training the most promising model(s)

Detail: Adaptive Learning Rate

Compute moving average of Δw_i^2 at time t for window length rho:

$$E[\Delta w_i^2]_t = rho * E[\Delta w_i^2]_{t-1} + (1-rho) * \Delta w_i^2$$

Compute RMS of Δw_i at time t with smoothing epsilon:

$$RMS[\Delta w_i]_t = sqrt(E[\Delta w_i^2]_t + epsilon)$$

Do the same for $\partial E/\partial w_i$, then obtain per-weight learning rate:

$$rate(w_i, t) = \frac{RMS[\Delta w_i]_{t-1}}{RMS[\partial E/\partial w_i]_t}$$

Adaptive acceleration / momentum: accumulate previous weight updates, but over a window of time

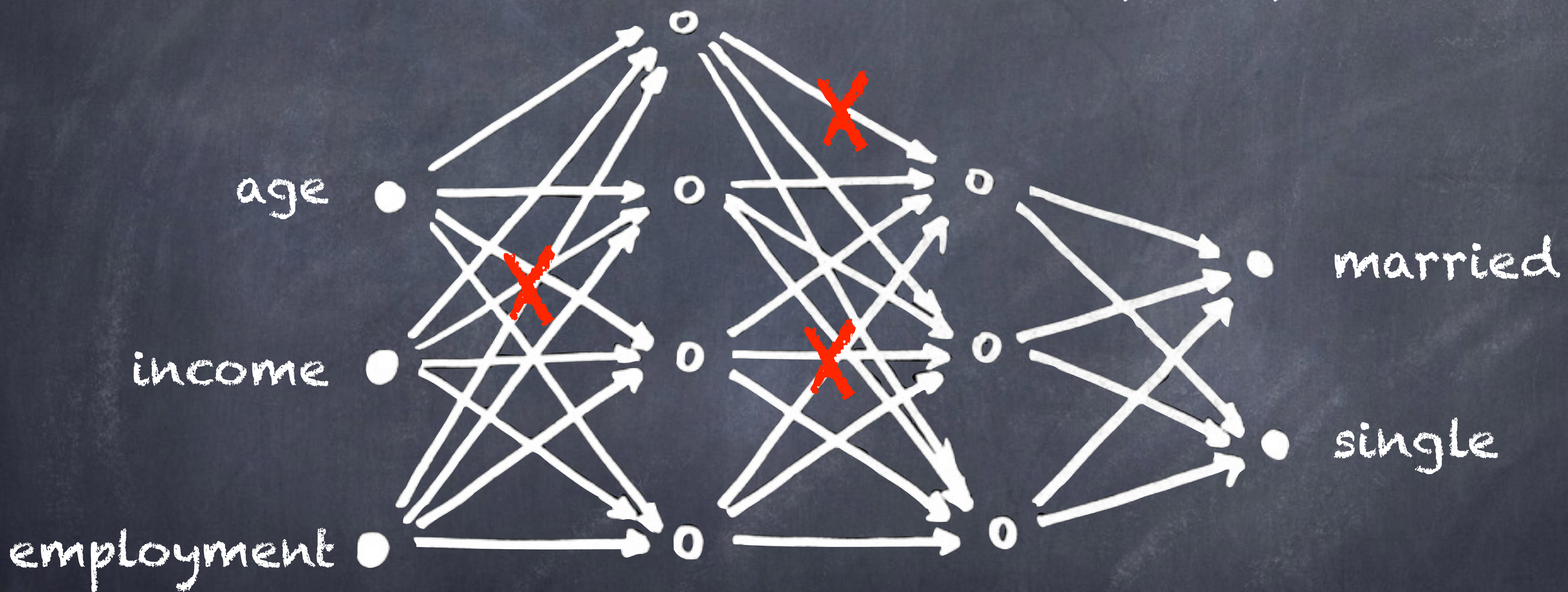
Adaptive annealing / progress: Gradient-dependent learning rate, moving window prevents "freezing" (unlike ADAGRAD: no window)

cf. ADADELTA paper

Detail: Dropout Regularization

Training:

For each hidden neuron, for each training sample, for each iteration, ignore (zero out) a different random fraction p of input activations.



Testing:

Use all activations, but reduce them by a factor p (to "simulate" the missing activations during training).

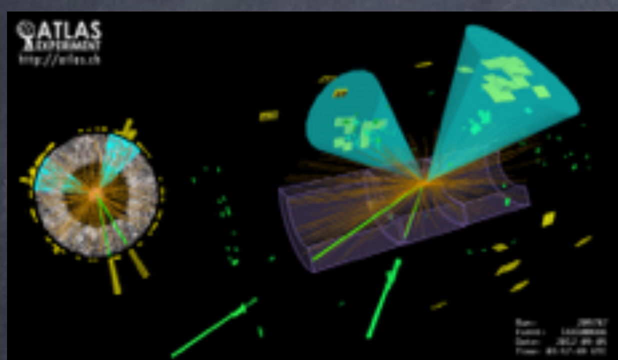
cf. Geoff Hinton's paper

Application: Higgs Boson Classification

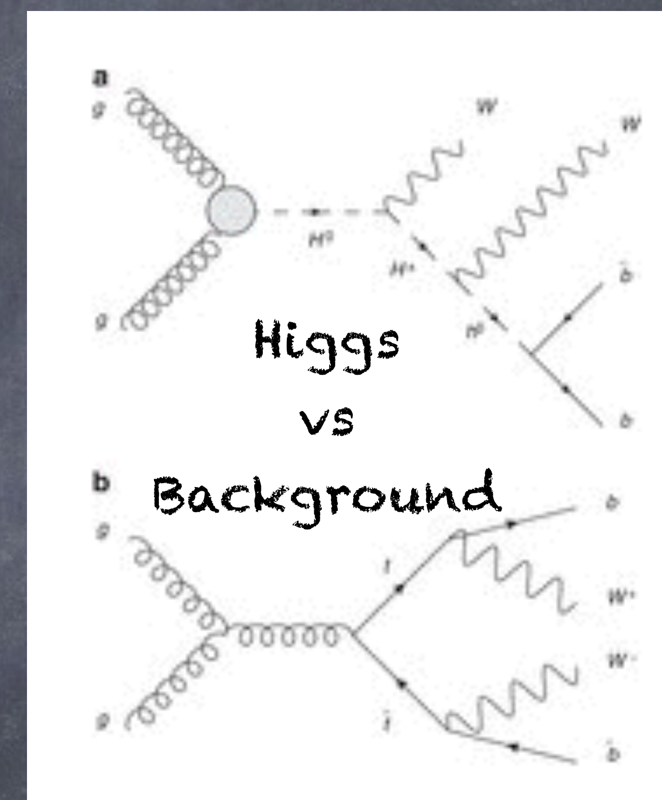
Large Hadron Collider: Largest experiment of mankind!

$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



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

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

Higgs: Derived features are important!

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

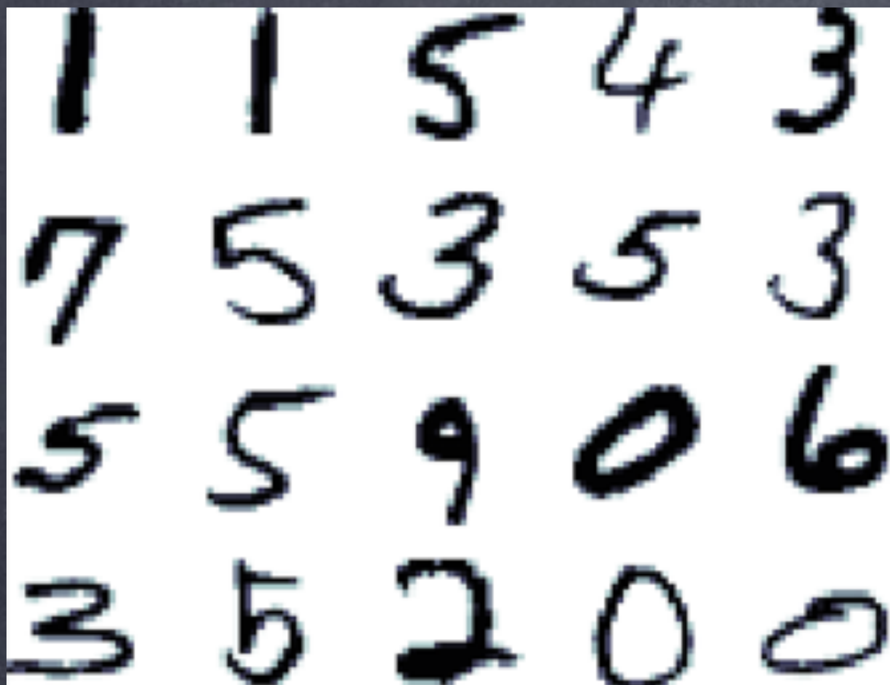
H2O Algorithm	low-level H2O AUC	all features H2O AUC
Generalized Linear Model	0.596	0.684
Random Forest	0.764	0.840
Gradient Boosted Trees	0.753	0.839
Neural Net 1 hidden layer	0.760	0.830
H2O Deep Learning	?	

add derived features

H2O

Live Demo: Let's see what Deep Learning can do with low-level features alone!

MNIST: digits classification



MNIST = Digitized handwritten digits database (Yann LeCun)

Yann LeCun: "Yet another advice: don't get fooled by people who claim to have a solution to Artificial General Intelligence. Ask them what error rate they get on MNIST or ImageNet."

Data: 28x28=784 pixels with (gray-scale) values in 0...255

Train:	60,000 rows	784 integer columns	10 classes
Test:	10,000 rows	784 integer columns	10 classes

Standing world record:
Without distortions or convolutions, the best-ever published error rate on test set: 0.83% (Microsoft)

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%

H2O Deep Learning beats MNIST

```
> install.packages("h2o")
> 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")
> record_model <- h2o.deeplearning(x = 1:784, y = 785, data = train_hex, validation = test_hex,
                                   activation = "RectifierWithDropout", hidden = c(1024,1024,2048),
                                   epochs = 8000, l1 = 1e-5, input_dropout_ratio = 0.2,
                                   train_samples_per_iteration = -1, classification_stop = -1)
|==========================================================================================================
===================================| 100%
> record_model@model$confusion
```

	Predicted										
Actual	0	1	2	3	4	5	6	7	8	9	Error
0	974	1	1	0	0	0	2	1	1	0	0.00612
1	0	1135	0	1	0	0	0	0	0	0	0.00088
2	0	0	1028	0	1	0	0	3	0	0	0.00388
3	0	0	1	1003	0	0	0	3	2	1	0.00693
4	0	0	1	0	971	0	4	0	0	6	0.01120
5	2	0	0	5	0	882	1	1	1	0	0.01121
6	2	3	0	1	1	2	949	0	0	0	0.00939
7	1	2	6	0	0	0	0	1019	0	0	0.00875
8	1	0	1	3	0	4	0	2	960	3	0.01437
9	1	2	0	0	4	3	0	2	0	997	0.01189
Totals	981	1142	1038	1013	977	891	956	1031	964	1007	0.00830

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

10 hours on 10 16-core nodes

World-record!

0.83% test set error

http://learn.h2o.ai/content/hands-on_training/deep_learning.html

POJO Model Export for Production Scoring

Java Model

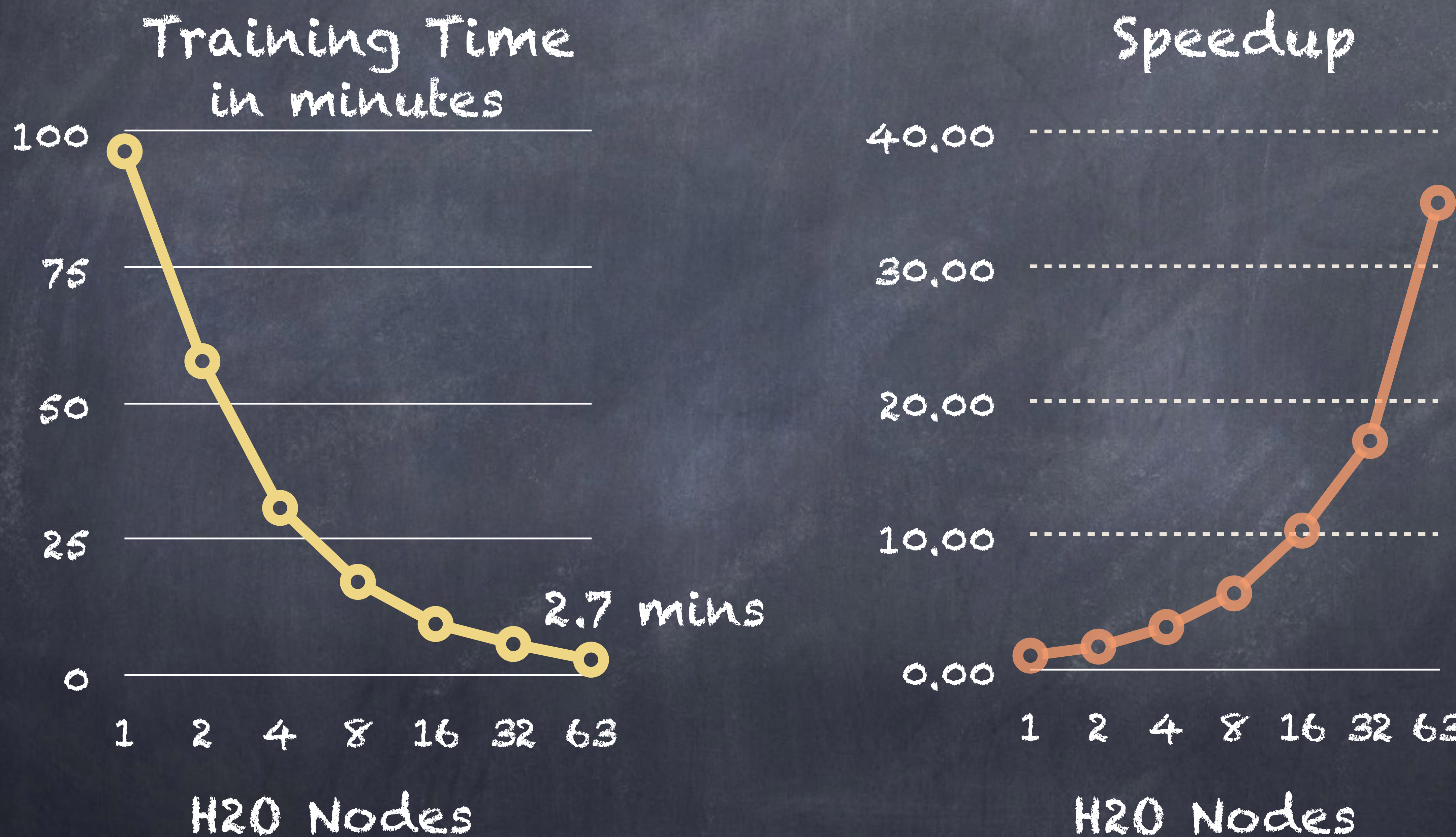
```java
import java.util.Map;
import water.genmodel.GenUtils.*;

// AUTOGENERATED BY H2O at Wed Jul 16 01:17:11 CDT 2014
// H2O v2.5.0.99999 (stable - 39e9d61cb3e33632a4bf21270ba48eeff601bdf6)
//
// Standalone prediction code with sample test data for DeepLearningModel named DeepLe
arning_8e3aca208af3a5ccee8ec8c697364564
//
// How to download, compile and execute:
//     mkdir tmpdir
//     cd tmpdir
//     curl http://127.0.0.1:54321/h2o-model.jar > h2o-model.jar
//     curl http://127.0.0.1:54321/2/DeepLearningModelView.java?_modelKey=De
_8e3aca208af3a5ccee8ec8c697364564 > DeepLearning_8e3aca208af3a5ccee8ec8c6973
//     javac -cp h2o-model.jar -J-Xmx2g -J-XX:MaxPermSize=128m DeepLearning_
f3a5ccee8ec8c697364564.java
//
//     (Note:  Try java argument -XX:+PrintCompilation to show runtime JIT c
havior.)

public class DeepLearning_8e3aca208af3a5ccee8ec8c697364564 extends water.gen
ratedModel {
  // Workspace for storing numerical input variables.
  public static final double[] NUMS = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
,0.0,0.0,0.0};
  // Standardization/Normalization scaling factor for numerical variables.
  public static final double[] NORMMUL = {0.28916567034872265,0.077086952828
```

Plain old Java code is auto-generated to take your H2O Deep Learning models into production!

```java
    for (i=0; i<ncats; ++i) ACTIVATION[0][CATS[i]] = 1f;
    for (i=0; i<NUMS.length; ++i) {
      ACTIVATION[0][CATOFFSETS[CATOFFSETS.length-1] + i] = Double.isNaN(NUMS[i]) ? 0f
: (float) NUMS[i];
    }
    for (i=1; i<ACTIVATION.length; ++i) {
      java.util.Arrays.fill(ACTIVATION[i],0f);
      for (int r=0; r<ACTIVATION[i].length; ++r) {
        final int cols = ACTIVATION[i-1].length;
        for (int c=0; c<cols; ++c) {
          ACTIVATION[i][r] += ACTIVATION[i-1][c] * WEIGHT[i][r*cols+c];
        }
        ACTIVATION[i][r] += BIAS[i][r];
      }
      if (i<ACTIVATION.length-1) {
        for (int r=0; r<ACTIVATION[i].length; ++r) {
          ACTIVATION[i][r] = 1f - 2f / (1f + (float)Math.exp(2*ACTIVATION[i][r]));
        }
      }
```

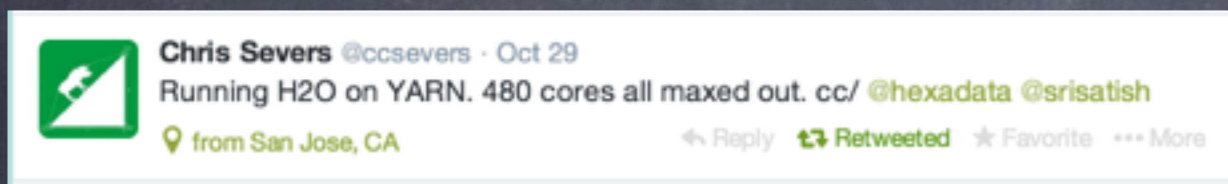
Parallel Scalability

(for 64 epochs on MNIST, with "0.83%" parameters)



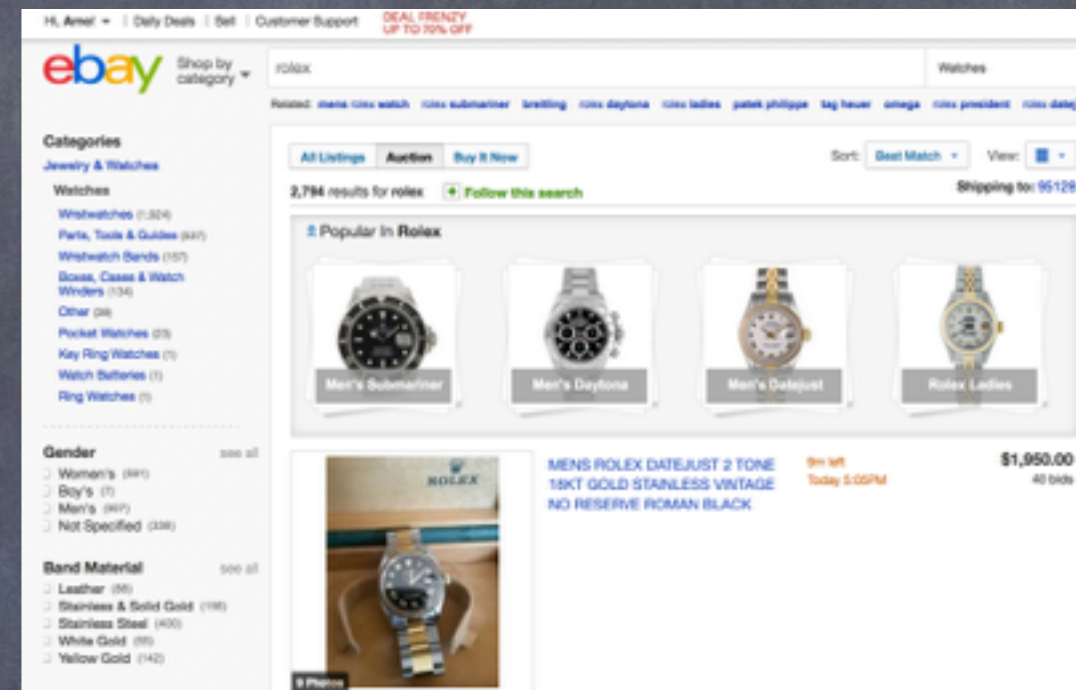
(4 cores per node, 1 epoch per node per MapReduce)

Text Classification



Goal: Predict the item from seller's text description

"Vintage 18KT gold Rolex 2 Tone in great condition"



Data: Bag of words vector 0,0,1,0,0,0,0,0,1,0,0,0,1,…,0

gold vintage condition

Train: 578,361 rows 8,647 cols 467 classes

Test: 64,263 rows 8,647 cols 143 classes

Text Classification

Train: 578,361 rows 8,647 cols 467 classes
Test: 64,263 rows 8,647 cols 143 classes

Out-Of-The-Box: 11.6% test set error after 10 epochs!
Predicts the correct class (out of 143) 88.4% of the time!

Classification error on training data: 9.78 %

Classification error on validation data: 11.60 %

Training samples: 7,000,151

Epochs: 10.005 / 10.000

Number of compute nodes: 4 (128 threads)

Training samples per iteration: 20,000

Training speed: 686 samples/s

Training time: 2:49:50.694

Hit Ratio for Multi-Class Classification

(Frequency of actual class label to be among the top-K predicted class labels)

K	Hit Ratio
1	88.402%
2	93.369%
3	94.922%
4	95.814%
5	96.410%
6	96.821%
7	97.149%
8	97.381%
9	97.555%
10	97.725%


```
"activation": "RectifierWithDropout",
"hidden": [
  1024,
  1024,
  1024
],
"epochs": 10.0,
"train_samples_per_iteration": 20000,
"seed": 8776149139794113198,
"adaptive_rate": "true",
"rho": 0.99,
"epsilon": 1.0E-8,
"input_dropout_ratio": 0.2,
"hidden_dropout_ratios": [
  0.5,
  0.5,
  0.5
],
"l1": 1.0E-5,
"l2": 0.0,
"max_w2": "Infinity",
```


Note 1: H2O columnar-compressed in-memory store only needs 60 MB to store 5 billion values (dense CSV needs 18 GB)

Note 2: No tuning was done (results are for illustration only)

MNIST: Unsupervised Anomaly Detection with Deep Learning (Autoencoder)

```
ae_model <- h2o.deeplearning(x=predictors,
                             y=42, #response (ignored - pick any non-constant column)
                             data=train_hex,
                             activation="Tanh",
                             autoencoder=T,
                             hidden=c(50),
                             ignore_const_cols=F,
                             epochs=1)

test_rec_error <- h2o.anomaly(test_hex, ae_model)
```

Download the script and run it yourself!

The good

The bad

The ugly

Higgs: Live Demo (Continued)

How well did Deep Learning do?

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?>		

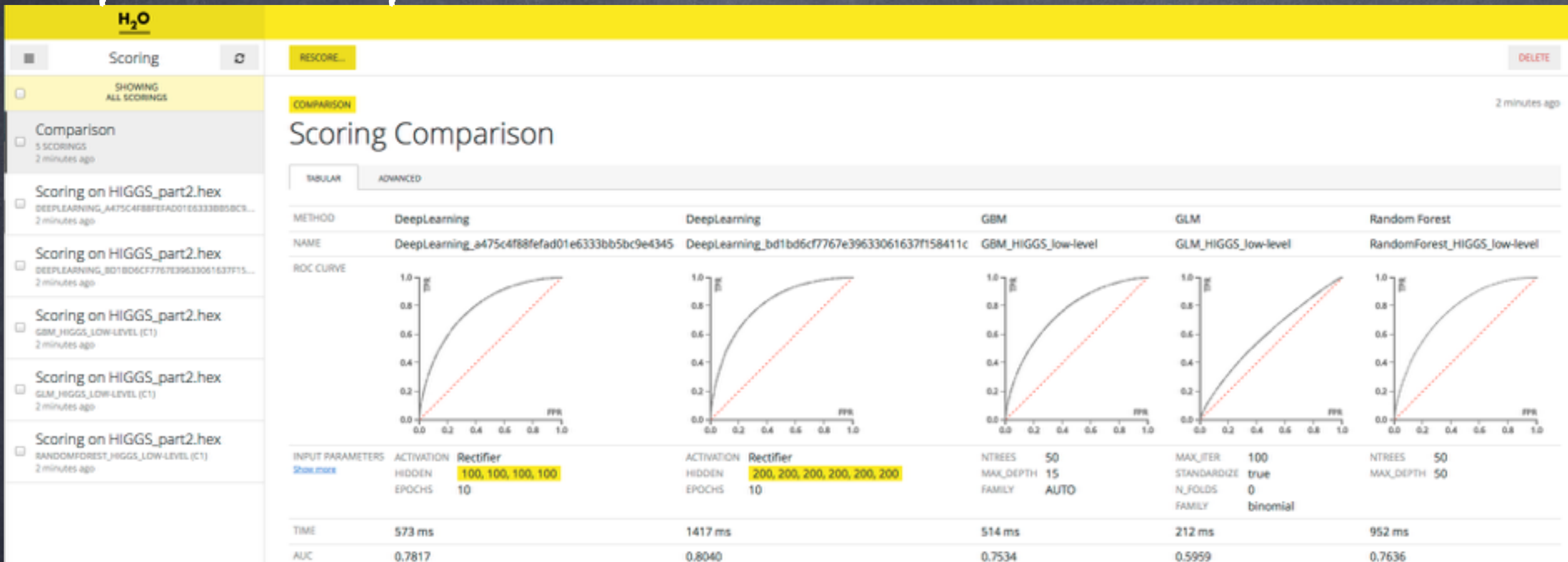
reference paper results

Any guesses for AUC on low-level features?
AUC=0.76 was the best for RF/GBM/NN (H2O)

Let's see how H2O did in the past 10 minutes!

H_2O Steam: Scoring Platform

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

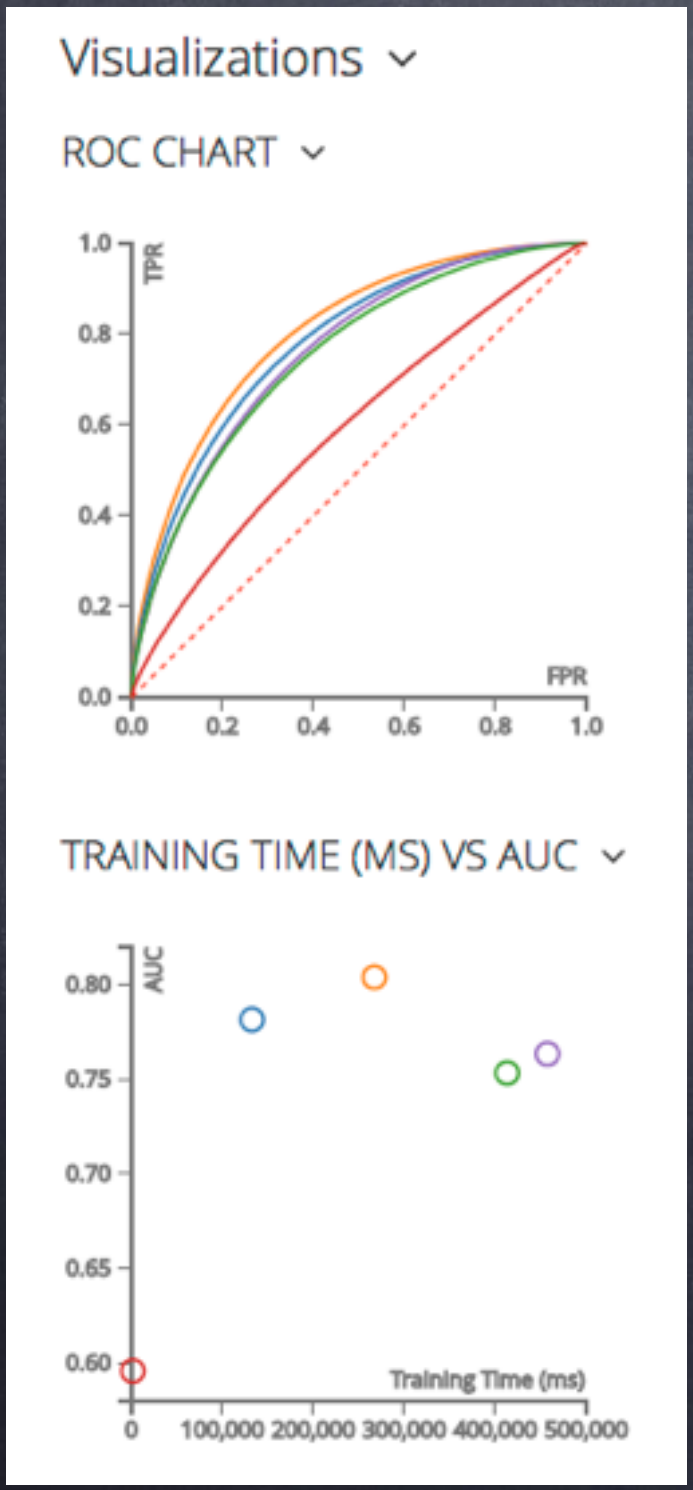


Higgs Dataset Demo on 10-node cluster

Let's score all our H_2O models and compare them!

Live Demo

Scoring Higgs Models in H2O Steam



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 H2O algos!
Better than LHC baseline of AUC=0.73!

Higgs Particle Detection with H2O

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)

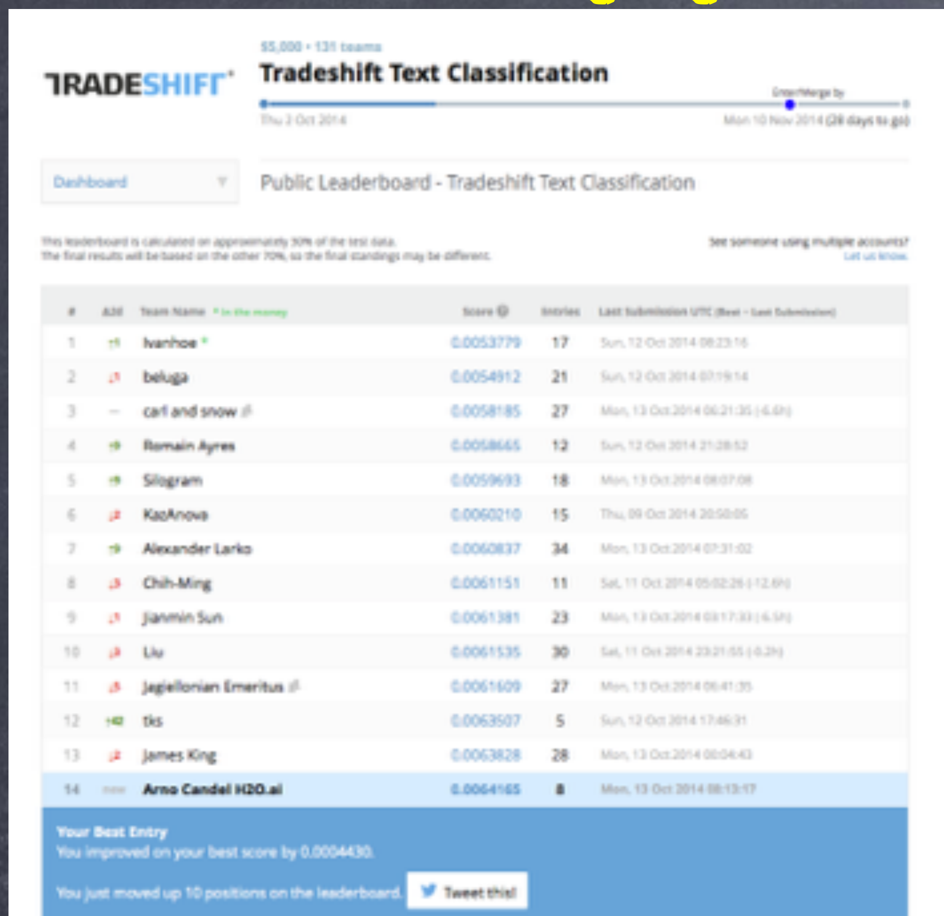
*nature paper: http://arxiv.org/pdf/1402.4735v2.pdf

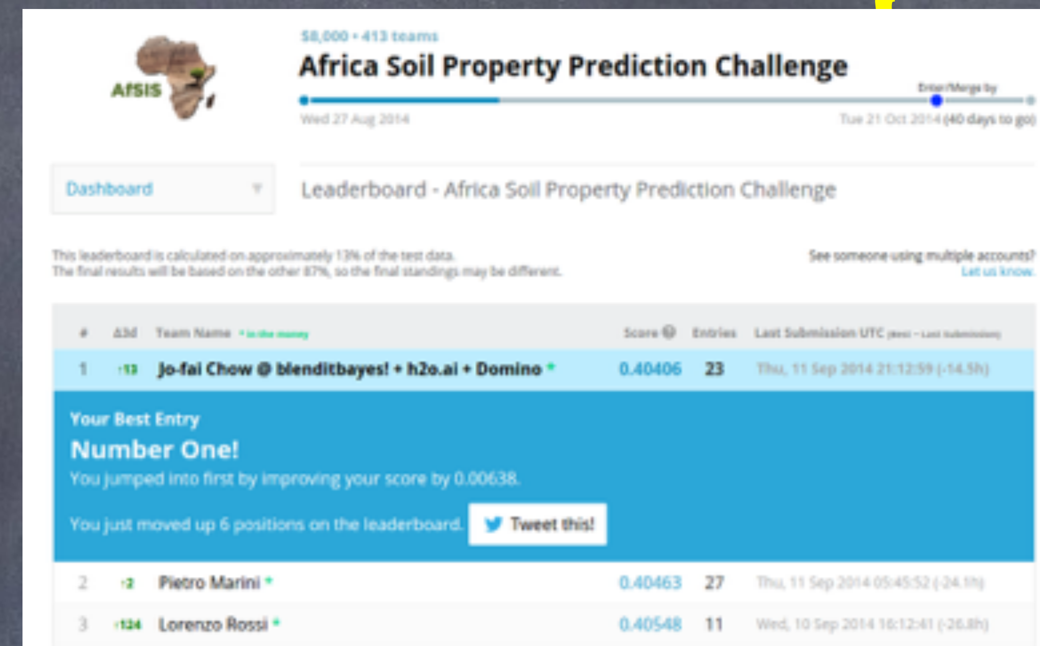
Algorithm	Paper's* L-L AUC	low-level H2O AUC	all features H2O AUC	Parameters (not heavily tuned), H2O running on 10 nodes
Generalized Linear Model	-	0.596	0.684	default, binomial
Random Forest	-	0.764	0.840	50 trees, max depth 50
Gradient Boosted Trees	0.73	0.753	0.839	50 trees, max depth 15
Neural Net 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 5 hidden layers	0.880	0.871	-	5x500 Rectifier, L1=L2=1e-5

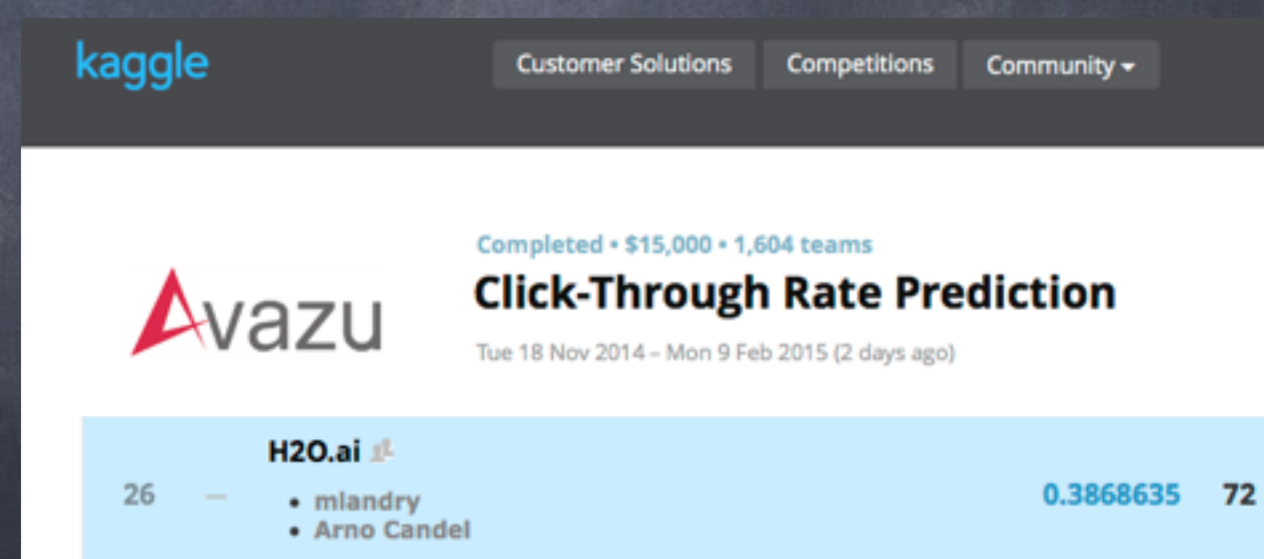
Deep Learning on low-level features alone beats everything else!

Prelim. H2O results compare well with paper's results* (TMVA & Theano)

H2O Kaggle Starter R Scripts



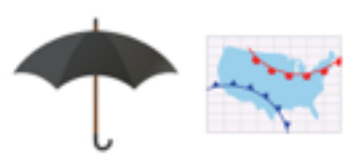




Final ranking:
#26 out of 1604

Currently Ongoing Challenge

kaggle — Customer Solutions | Competitions | Community ▾ — Arno Candel | Logout



$500 • 58 teams

How much did it rain?

Enter/Merge by

Fri 9 Jan 2015 — Fri 15 May 2015 (3 months to go)

Dashboard

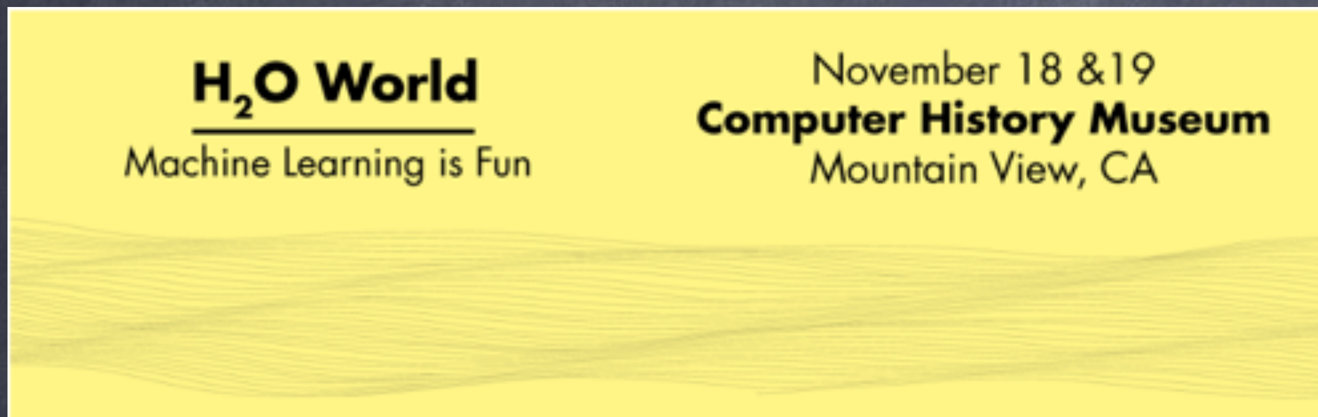
Public Leaderboard - How much did it rain?

This leaderboard is calculated on approximately 70% of the test data.
The final results will be based on the other 30%, so the final standings may be different.

See someone using multiple accounts?
Let us know.

#	Δ6d	Team Name • in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑13	H2O.ai	0.00793891	5	Tue, 03 Feb 2015 01:56:29
2	↑8	vrajs5	0.00801716	9	Thu, 05 Feb 2015 03:57:48 (-47.9h)
3	new	utility	0.00808444	5	Thu, 05 Feb 2015 16:02:07
4	↓3	Jamounel	0.00815678	15	Thu, 05 Feb 2015 23:44:45 (-35.9h)
5	↓3	StormMiner	0.00838693	3	Thu, 05 Feb 2015 00:26:08 (-16.3d)
6	↓3	khyh	0.00844691	4	Sun, 01 Feb 2015 05:29:54 (-39.1h)
7	↑4	Roberto	0.00861345	5	Wed, 04 Feb 2015 18:20:33

Re-Live H2O World!



http://h2o.ai/h2o-world/

http://learn.h2o.ai

Watch the Videos

Day 1

- Hands-On Training
  - Supervised
  - Unsupervised
  - Advanced Topics
  - Markting Usecase
- Product Demos
- Hacker-Fest with Cliff Click (CTO, Hotspot)

Day 2

- Speakers from Academia & Industry
  - Trevor Hastie (ML)
  - John Chambers (S, R)
  - Josh Bloch (Java API)
  - Many use cases from customers
  - 3 Top Kaggle Contestants (Top 10)
- 3 Panel discussions

H2O GitBooks

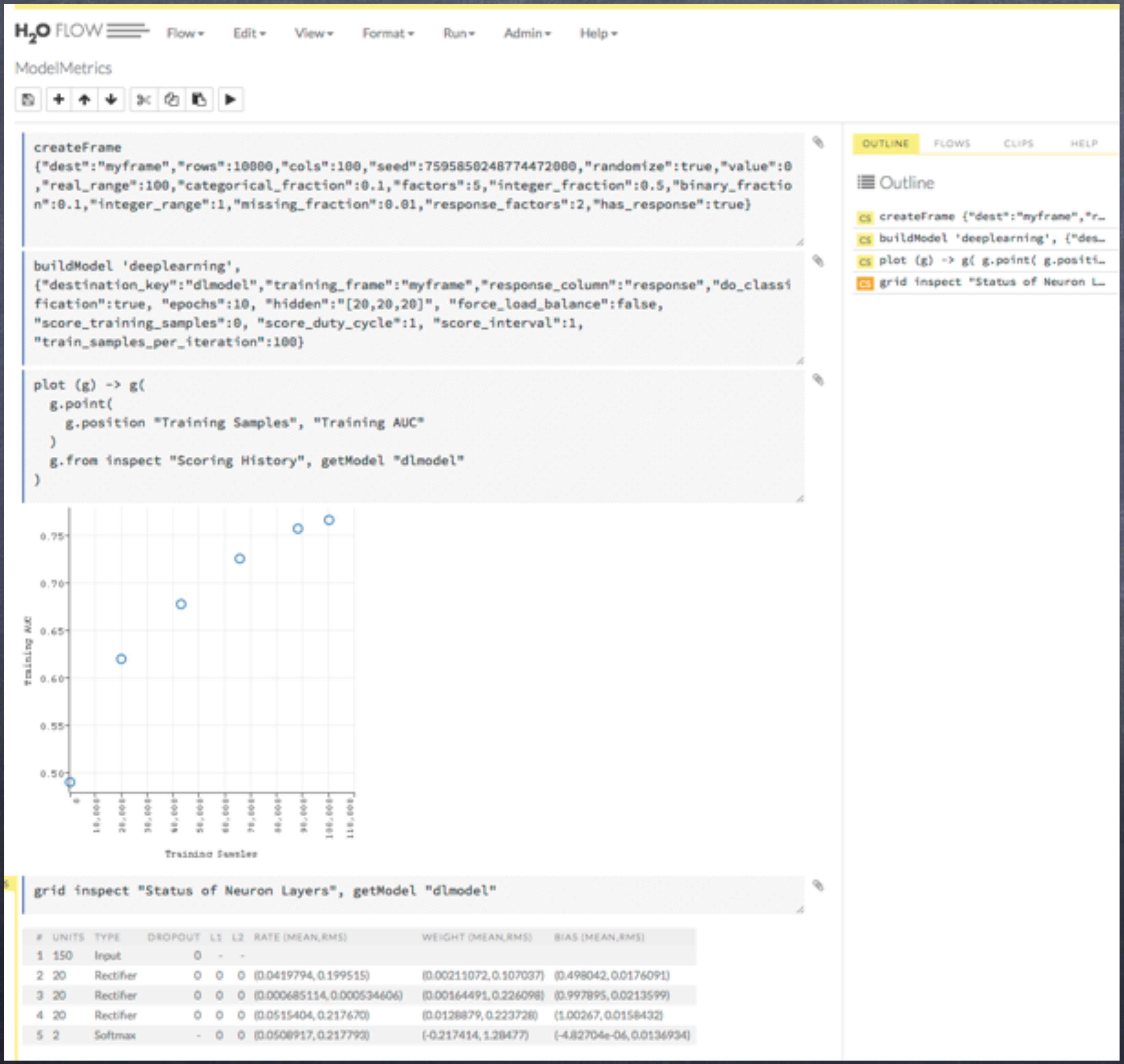
H2O World
learn.h2o.ai

R, EC2,
Hadoop

Deep
Learning

Also available: GBM & GLM GitBooks
at http://h2o.gitbooks.io

h2o-dev: Flow, Python, JS, Spark



h2o-dev Python Example

```python
import sys
sys.path.insert(1, "..")
import h2o


def deep_learning_metrics_test(ip, port):
  h2o.init(ip, port)             # connect to existing cluster
  df = h2o.import_frame(path="smalldata/logreg/prostate.csv")

  del df['ID']                                    # remove ID
  df['CAPSULE'] = df['CAPSULE'].asfactor()        # make CAPSULE categorical
  vol = df['VOL']
  vol[vol == 0] = None                            # 0 VOL means 'missing'

  r = vol.runif()                                 # random train/test split
  train = df[r < 0.8]
  test  = df[r >= 0.8]

  # See that the data is ready
  train.describe()
  train.head()
  test.describe()
  test.head()

  # Run DeepLearning

  print "Train a Deeplearning model: "
  dl = h2o.deeplearning(x           = train[1:],
                        y           = train['CAPSULE'],
                        epochs = 100,
                        hidden = [10, 10, 10])
  print "Binomial Model Metrics: "
  print
  dl.show()
  # print dl._model_json
  dl.model_performance(test).show()


if __name__ == "__main__":
  args = sys.argv
  print args
  if len(args) > 1:   deep_learning_metrics_test(args[1],int(args[2]))
  else:               deep_learning_metrics_test("localhost",54321)
```

You can participate!

- Images: Convolutional & Pooling Layers PUB-644
- Sequences: Recurrent Neural Networks PUB-1052
- Faster Training: GPGPU support PUB-1013
- Pre-Training: Stacked Auto-Encoders PUB-1014
- Ensembles PUB-1072
- Use H2O at Kaggle Challenges!

Key Take-Aways

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

H2O Deep Learning is ready to take your advanced analytics to the next level - Try it on your data!

Join our Community and Meetups!

- https://github.com/h2oai
- h2ostream community forum
- www.h2o.ai
- @h2oai

Thank you!