Deep Learning for Internet of Things Application Using H2O Platform

Basheer Qolomany

CS6030: Internet of Things – Application Development
Internet of Things (IoT) is heavily signal data

Information from the Internet of Things:
We have gone beyond the decimal system

Today data scientist uses Yottabytes to describe how much government data the NSA or FBI have on people altogether.

In the near future, Brontobyte will be the measurement to describe the type of sensor data that will be generated from the IoT (Internet of Things).

Yottabyte
This is our digital universe today
= 250 trillion of DVDs

1 EB of data is created on the internet each day = 250 million DVDs worth of information. The proposed Square Kilometer Array telescope will generate an EB of data per day.

Exabyte
10^{24}

Zettabyte
1.3 ZB of network traffic by 2016

Brontobyte
This will be our digital universe tomorrow...

Petabyte
The CERN Large Hadron Collider generates 1PB per second.

Terabyte
500TB of new data per day are ingested in Facebook databases.

Gigabyte

Megabyte

Yottabyte

Exabyte
Machine Learning - Definition

• A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data.

• “The ability of a program to learn from experience—that is, to modify its execution on the basis of newly acquired information.”
What is Clustering?

- **Clustering**: is the assignment of a set of observations into subsets (called *clusters*) so that observations in the same cluster are similar in some sense. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis used in many fields.
What is Classification?

- **Classification** is the task of *learning a target function* \( f \) that maps attribute set \( x \) to one of the predefined class labels \( y \).

One of the attributes is the class attribute.
In this case: Cheat

Two class labels (or classes): Yes (1), No (0)

![Figure 4.2](image_url)
**Catching tax-evasion**

Tax-return data for year 2011

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

A new tax return for 2012
Is this a cheating tax return?

<table>
<thead>
<tr>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Married</td>
<td>80K</td>
<td>?</td>
</tr>
</tbody>
</table>

An instance of the classification problem: learn a method for discriminating between records of different classes (cheaters vs non-cheaters)
Why Classification?

- The target function $f$ is known as a classification model.

- **Descriptive modeling:** Explanatory tool to distinguish between objects of different classes (e.g., understand why people cheat on their taxes).

- **Predictive modeling:** Predict a class of a previously unseen record.
Illustrating Classification Task

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Large</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Medium</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Small</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Medium</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Large</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Medium</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Large</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Small</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Medium</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Small</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tid</th>
<th>Attr1</th>
<th>Attr2</th>
<th>Attr3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No</td>
<td>Small</td>
<td>55K</td>
<td>?</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Medium</td>
<td>80K</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Large</td>
<td>110K</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Small</td>
<td>95K</td>
<td>?</td>
</tr>
<tr>
<td>15</td>
<td>No</td>
<td>Large</td>
<td>67K</td>
<td>?</td>
</tr>
</tbody>
</table>
In classification, you first 'Learn' what goes with what and then you 'Apply' that knowledge to new examples. So if somebody gave us the first picture on the left, which is a plot of hair length (Y axis) against gender (on X axis),

In this case, clustering algorithm has to "Infer" that you could create at least two groups of points.
The curse of dimensionality

- Real data usually have **thousands**, or **millions** of dimensions
  - E.g., web documents, where the dimensionality is the vocabulary of words
  - Facebook graph, where the dimensionality is the number of users
- Huge number of dimensions causes problems
  - Data becomes very **sparse**, some algorithms become meaningless (e.g. density based clustering)
  - The **complexity** of several algorithms depends on the dimensionality and they become infeasible.
Dimensionality reduction

• In machine learning and statistics, dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration, via obtaining a set of "uncorrelated" principal variables.
• Usually the data can be described with fewer dimensions, without losing much of the meaning of the data.
  ▪ Essentially, we assume that some of the data is noise, and we can approximate the useful part with a lower dimensionality space.
    ▪ Dimensionality reduction does not just reduce the amount of data, it often brings out the useful part of the data.
What is Deep Learning?

- it’s a computer algorithm that models high-level abstractions in data with multiple layers of non-linear transformations.
<table>
<thead>
<tr>
<th>Gartner Tech Trends</th>
<th>Description</th>
<th>Relates to this talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Machine Learning</td>
<td>Deep Neural Nets</td>
<td>DNN to solve application needs</td>
</tr>
<tr>
<td>Device Mesh</td>
<td>Mobile, wearable, home, auto, IoT</td>
<td>Practical applications, input data</td>
</tr>
<tr>
<td>Adaptive Security Architecture</td>
<td>Move from static rules and patterns to understand user and systems</td>
<td>Practical applications</td>
</tr>
<tr>
<td>Information of Everything</td>
<td>Contextual, integrated</td>
<td>Input data</td>
</tr>
<tr>
<td>Ambient User Experience</td>
<td>Over environments, time location</td>
<td>Output to users (i.e. real time scoring)</td>
</tr>
<tr>
<td>Autonomous Agents and Things</td>
<td>Smart advisors</td>
<td>Output to users (i.e. real time scoring)</td>
</tr>
<tr>
<td>Advanced System Architectures</td>
<td>Train DNN with GPUs and FPGAs, cloud architectures</td>
<td>Train DNN</td>
</tr>
<tr>
<td>Mesh App and Service Architecture</td>
<td>3 tier --&gt; loosely coupled apps and services for web scale performance &amp; flexibility</td>
<td>Application deployment architecture</td>
</tr>
<tr>
<td>Internet of Things Platforms</td>
<td>Complements mesh app and service arch, implementations of IoT</td>
<td>Input data system integrating with deployment architecture</td>
</tr>
<tr>
<td>3D Printing</td>
<td>Expect annual growth rate of 64% for enterprise printers through 2019</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Gartner Identifies the Top 10 Internet of Things Technologies for 2017 and 2018

- IoT Security
- IoT Analytics
- IoT Device Management
- Low-Power, Short-Range IoT Networks
- Low-Power, Wide-Area Networks
- IoT Processors
- IoT Operating Systems
- Event Stream Processing
- IoT Platforms
- IoT Standards and Ecosystems

Source: Gartner Identifies the Top 10 Internet of Things Technologies for 2017 and 2018
Published February 23, 2016
What problems can deep machine learning address?

- Spam Detection
- Credit Card Fraud Detection
- Digit Recognition
- Speech Understanding
- Face Detection
- Product Recommendation
- Medical Diagnosis
- Stock Trading
- Customer Segmentation
- Shape Detection
Step 1: Great Algorithms + Fast Computers

- Raw computing power can automate complex tasks!

1997: Playing Chess
(IBM Deep Blue beats Kasparov)

Earlier he said: “No computer will ever beat me.”

Computer Science
30 custom CPUs, 60 billion moves in 3 mins
Step 2: More Data + Real-Time Processing

• Automating automobiles into autonomous automata!

2005: Self-driving Cars
DARPA Grand Challenge, 132 miles (won by Stanford A.I. lab*)

Sensors & Computer Science
video, radar, laser, GPS, 7 Pentium computers
Step 3: Big Data + In-Memory Clusters

- Automating question answering and information retrieval!

Note: IBM Watson received the question in electronic written form, and was often able to (electronically) press the answer button faster than the competing humans.
Step 4: Deep Learning

- Deep Learning + Smart Algorithms = Master Gamer.

2014: Atari Games (DeepMind)
trained from raw pixel values, no human rules

Deep Learning
+ reinforcement learning, tree search,
Monte Carlo, GPUs, playing against itself, ...

2016: AlphaGo (Google DeepMind)
Step 5: Improve Training Efficiency

• New algorithm learns handwriting of unseen symbols from very few training examples (unlike typical Deep Learning).

2015: MIT, NYU, Toronto

Bayesian Program Learning
(NOT Deep Learning)
What ELSE can Deep Learning do?

- Deep Learning can generate handwriting

I promise not to eat your lunch. I promise not to eat your lunch. I promise not to eat your lunch.
What ELSE can Deep Learning do?

- Deep Learning can generate code, captions, language, etc.

- Generated math proof:

\[
\begin{align*}
\text{Spec}(K_e) 
\xrightarrow{\alpha'} 
\xrightarrow{\alpha} 
\end{align*}
\]

is a limit. Then \( \mathcal{G} \) is a finite type and assume \( S \) is a flat and \( \mathcal{F} \) and \( \mathcal{G} \) is a finite type \( f_* \). This is of finite type diagrams, and

- the composition of \( \mathcal{G} \) is a regular sequence,
- \( \mathcal{O}_{X'} \) is a sheaf of rings.

\[\text{Proof.} \ \text{We have see that} \ X = \text{Spec}(R) \ \text{and} \ \mathcal{F} \ \text{is a finite type representable by algebraic space. The property} \ \mathcal{F} \ \text{is a finite morphism of algebraic stacks. Then the cohomology of} \ X \ \text{is an open neighbourhood of} \ U.\]

\[\text{Proof.} \ \text{This is clear that} \ \mathcal{G} \ \text{is a finite presentation, see Lemmas ??}. \ A \text{ reduced above we conclude that} \ U \ \text{is an open covering of} \ \mathcal{C}. \ \text{The functor} \ \mathcal{F} \ \text{is a} \ \text{“field} \]

\[
\mathcal{O}_{X,x} \xrightarrow{\mathcal{F}_x} \mathcal{F}_x^{-1}(\mathcal{O}_{X,x}) \xrightarrow{\mathcal{O}_{X,x}^{-1}} \mathcal{O}_{X,x}(\mathcal{O}_{X,x}^\mathcal{F})
\]

is an isomorphism of covering of \( \mathcal{O}_{X,x} \). If \( \mathcal{F} \) is the unique element of \( \mathcal{F} \) such that \( X \) is an isomorphism.

Image captioning:

“black and white dog jumps over bar.”
What ELSE can Deep Learning do?

- Deep Learning can translate any language
What ELSE can Deep Learning do?

• Deep Learning can create masterpieces: Semantic Style Transfer
# Deep Learning Tools

<table>
<thead>
<tr>
<th>System or Package</th>
<th>Language</th>
<th>Algorithms and Comments</th>
<th>Org Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theano</td>
<td>Python, NumPy scripting</td>
<td>Python expressions in Theano, can compile to C with g++, use GPU (CUDA), parallel execution. Oldest and</td>
<td>Univ of Montreal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>most used mature deep learning system.</td>
<td></td>
</tr>
<tr>
<td>Torch7</td>
<td>Lua scripting, command line interpreter</td>
<td>N-dimensional array or Tensor. Used by Google DeepMind, Facebook AI, IBM, Yandex. On Android and iOS</td>
<td></td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Python scripting, C++ back</td>
<td>Tensors + flow graphs. 2nd generation deep learning at Google, just came out Nov 2015. CPU, GPU &amp; mobile.</td>
<td>Google</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integrated network GUI. Jeff Dean</td>
<td></td>
</tr>
</tbody>
</table>
Deep Learning Tools

By Google, 600 DL proj
Speech
Google Photos
Translation
Gmail
Search

Rajat Monga, Tech Lead & Manager for TensorFlow
# Deep Learning Tools

<table>
<thead>
<tr>
<th>System or Package</th>
<th>Language</th>
<th>Algorithms and Comments</th>
<th>Org Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2O</td>
<td>Java, Python, R, Scala, JSON</td>
<td>autoencoder, convolutional, parallel weight updates over servers</td>
<td>H2O.ai</td>
</tr>
<tr>
<td>DL4J DeepLearning4j</td>
<td>Java, Scala</td>
<td>autoencoder, deep belief network, DBN, word vector, recurrent net</td>
<td>Skymind</td>
</tr>
<tr>
<td>Gorila</td>
<td>General Reinforcement Learning Architecture, David Silver</td>
<td>Google DeepMind</td>
<td></td>
</tr>
<tr>
<td>Kamanja</td>
<td>18 data mining systems generate PMML. Also Java &amp; Scala</td>
<td>Various neural net training systems, such as R. Good compute framework for scaling up full system, end to end.</td>
<td>LigaDATA</td>
</tr>
<tr>
<td>deepnet</td>
<td>R</td>
<td>BP, RBM, DBN, autoencoder</td>
<td></td>
</tr>
<tr>
<td>darch</td>
<td>R</td>
<td>Restricted Boltzmann Machines (RBM)</td>
<td></td>
</tr>
<tr>
<td>autoencoder</td>
<td>R</td>
<td>Sparse autoencoder, from Ng's notes - for deep belief neural nets</td>
<td></td>
</tr>
</tbody>
</table>
## What is H2O?

### Math Platform
Open source in-memory prediction engine
- Parallelized and distributed algorithms making the most use out of multithreaded systems
- GLM, Random Forest, GBM, PCA, etc.

### API
Easy to use and adopt
- Written in Java – perfect for Java Programmers
- REST API (JSON) – drives H2O from R, Python, Java, Scala, Excel, Tableau

### Big Data
More data? Or better models? BOTH
- Use all of your data – model without down sampling
- Run a simple GLM or a more complex GBM to find the best fit for the data
- More Data + Better Models = Better Predictions
H2O Platform Overview

• Distributed implementations of cutting edge ML algorithms.
• Core algorithms written in high performance Java.
• APIs available in R, Python, Scala, REST/JSON.
• Interactive Web GUI.
H2O Platform Overview

• Write code in high-level language like R (or use the web GUI) and output production-ready models in Java.
• To scale, just add nodes to your H2O cluster.
• Works with Hadoop, Spark and your laptop.
H2O Production Analytics Workflow

H2O Compute Engine

Load Data
- Distributed In-Memory
- Loss-less Compression

Exploratory & Descriptive Analysis

Feature Engineering & Selection

Supervised & Unsupervised Modeling

Model Evaluation & Selection

Predict

Data & Model Storage

Data Prep Export: Plain Old Java Object

Model Export: Plain Old Java Object

Production Scoring Environment

Spark

kafka

STORM

HIVE

Your Imagination
Algorithms on H$_2$O

Supervised Learning

- **Generalized Linear Models with Regularization**: Binomial, Gaussian, Gamma, Poisson and Tweedie
- **Naïve Bayes**
- **Distributed Random Forest**: Classification or regression models
- **Gradient Boosting Machine**: Produces an ensemble of decision trees with increasing refined approximations
- **Deep learning**: Create multi-layer feed forward neural networks starting with an input layer followed by multiple layers of nonlinear transformations
Algorithms on H$_2$O

Unsupervised Learning

- **K-means**: Partitions observations into k clusters/groups of the same spatial size
- **Principal Component Analysis**: Linearly transforms correlated variables to independent components
- **Generalized Low Rank Models***: extend the idea of PCA to handle arbitrary data consisting of numerical, Boolean, categorical, and missing data
- **Autoencoders**: Find outliers using a nonlinear dimensionality reduction using deep learning
H2O Software Stack

Rapids Expression Evaluation Engine

Scala

Customer Algorithm

Fluid Vector Frame
Distributed K/V Store
Non-blocking Hash Map

Parse
GLM
GBM
RF
Deep Learning
K-Means
PCA

In-H2O Prediction Engine

Customer Algorithm

Job
MRTask
Fork/Join

Spark
Hadoop
Standalone H2O
## H2O Components

| H2O Cluster                                      | • Multi-node cluster with shared memory model.  
|                                                 | • All computations in memory.  
|                                                 | • Each node sees only some rows of the data.  
|                                                 | • No limit on cluster size.  
| Distributed Key Value Store                    | • Objects in the H2O cluster such as data frames, models and results are all referenced by key.  
|                                                 | • Any node in the cluster can access any object in the cluster by key.  
| H2O Frame                                       | • Distributed data frames (collection of vectors).  
|                                                 | • Columns are distributed (across nodes) arrays.  
|                                                 | • Each node must be able to see the entire dataset (achieved using HDFS, S3, or multiple copies of the data if it is a CSV file).  

Distributed K/V Store

- The H2O K/V Store is a classic peer-to-peer distributed hash table.
- There is no “name-node” nor central key dictionary.

- Each key has a home-node, but the homes are picked pseudo-randomly per-key.
- This allows us to force keys to “home” to different nodes (usually for load-balance reasons).
- A key's “home” is solely responsible for breaking ties in racing writes and is the “source of truth.”
- Keys can be cached anywhere, and both reads & writes can be cached (although a write is not complete until it reaches “home”.)
Data in H2O

- We read data fully parallelized from: HDFS, NFS, Amazon S3, URLs, URIs, CSV, SVMLight.
- Data is highly compressed (about 2-4 times smaller than gzip).

Highly Compressed

- Memory bound, not CPU bound.
- If data accessed linearly, as fast as C or Fortran.
- Speed = data volume / memory bandwidth
- ~50GB / sec (varies by hardware).

Speed

- Table width: <1k fast, <10k works, <100k slow
- Table length: Limited only by memory

Data Shape
What is R?

• The R statistical programming language is a free open source package based on the S language developed by Bell Labs.

• The language is very powerful for writing programs.

• Many statistical functions are already built in. It includes routines for data summary and exploration, graphical presentation and data modelling.

• Contributed packages expand the functionality to cutting edge research.

• Since it is a programming language, generating computer code to complete tasks is required.
How to download?

– Google it using R or CRAN
(Comprehensive R Archive Network)
– http://www.r-project.org
Getting Started

- The R GUI?
Getting Started

• Opening a script.
• This gives you a script window.
You can enter commands one at a time at the command prompt (>) or run a set of commands from a source file.

There is a wide variety of data types, including vectors (numerical, character, logical), matrices, dataframes, and lists.

To quit R, use

>q()
R Overview

• Basic assignment and operations.
• Arithmetic Operations:
  – +, -, *, /, ^ are the standard arithmetic operators.
• Matrix Arithmetic.
  – * is element wise multiplication
  – %*% is matrix multiplication
• Assignment
  – To assign a value to a variable use “<-”
R Overview

• If you know which function you want help with simply use `?functionname`

• At any time we can list the objects which we have created: `ls()`

• More commonly a function will operate on an object, for example: `sqrt(16)`

• Vectors can be created in R in a number of ways. We can describe all of the elements: `z<-c(5,9,1,0)`
R Overview

- Objects can be removed from the current workspace with the `rm` function:
  - `rm(z)`
- Sequences can be generated as follows: `x<-1:10`
- While more general sequences can be generated using the `seq` command. For example:
  - `seq(1,9,by=2)` or `seq(8,20,length=6)`
Matrices

- Matrices can be created in R in a variety of ways. Perhaps the simplest is to create the columns and then glue them together with the command `cbind`.
- `> x<-c(5,7,9)`
- `> y<-c(6,3,4)`
- `> z<-cbind(x,y)`
- `> z`
- The dimension of a matrix can be checked with the `dim` command:
  - `> dim(z)`
- Matrices can also be built by explicit construction via the `function matrix`. For example,
  - `z<-matrix(c(5,7,9,6,3,4),nrow=3)`
R Workspace

Objects that you create during an R session are held in memory, the collection of objects that you currently have is called the workspace. This workspace is not saved on disk unless you tell R to do so. This means that your objects are lost when you close R and not save the objects, or worse when R or your system crashes on you during a session.
R Workspace

# save your command history
savehistory(file="myfile")  # default is ".Rhistory"

# recall your command history
loadhistory(file="myfile")  # default is ".Rhistory"
R Datasets

R comes with a number of sample datasets that you can experiment with. Type

> data()

to see the available datasets. The results will depend on which packages you have loaded. Type

help(datasetname)

for details on a sample dataset.
When you download R, already a number (around 30) of packages are downloaded as well. To use a function in an R package, that package has to be attached to the system. When you start R not all of the downloaded packages are attached, only seven packages are attached to the system by default. You can use the function search to see a list of packages that are currently attached to the system, this list is also called the search path.

```r
> search()
[1] ".GlobalEnv" "package:stats" "package:graphics"
```
### “h2o” R package on CRAN

#### Requirements
- The only requirement to run the “h2o” R package is R >=3.1.0 and Java 7 or later.
- Tested on many versions of Linux, OS X and Windows.

#### Installation
- The easiest way to install the “h2o” R package is to install directly from CRAN.
- Latest version: [http://h2o.ai/download](http://h2o.ai/download)

#### Design
- No computation is ever performed in R.
- All computations are performed (in highly optimized Java code) in the H2O cluster and initiated by REST calls from R.
Start H2O  Cluster from R

```r
> library(h2o)
> localH2O <- h2o.init(nthreads = -1, max_mem_size = "8G")

H2O is not running yet, starting it now...

Note: In case of errors look at the following log files:
   /var/folders/2j/jg4s153d5q53tc2_nzm9fz5h00000gn/T//RtmpAXY9gj/h2o_me_started_from_r.out
   /var/folders/2j/jg4s153d5q53tc2_nzm9fz5h00000gn/T//RtmpAXY9gj/h2o_me_started_from_r.err

java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

Successfully connected to http://127.0.0.1:54321/

R is connected to the H2O cluster:
  H2O cluster uptime: 1 seconds 96 milliseconds
  H2O cluster version: 3.3.0.99999
  H2O cluster name: H2O_started_from_R_me_kfo618
  H2O cluster total nodes: 1
  H2O cluster total memory: 7.11 GB
  H2O cluster total cores: 8
  H2O cluster allowed cores: 8
  H2O cluster healthy: TRUE

> 
```
H2O in R: Load Data

Example

```r
library(h2o)  # First install from CRAN
localH2O <- h2o.init()  # Initialize the H2O cluster

# Data directly into H2O cluster (avoids R)
train <- h2o.importFile(path = "train.csv")

# Data into H2O from R data.frame
train <- as.h2o(my_df)
```

R code example: Load data
Reading Data from HDFS into H2O with R

STEP 2

2.1 R function call

2.2 h2o.importFile() has HDFS path

2.3 Initiate distributed ingest

2.4 Request data from HDFS

H2O Cluster

HDFS

data.csv
Reading Data from HDFS into H2O with R

**STEP 3**

1. **R**
   - h2o_df
     - Cluster IP
     - Cluster Port
     - Pointer to Data

2. **HDFS**
   - data.csv
     - HDFS provides data

3. **H2O Cluster**
   - Distributed H2O Frame in DKV

4. **H2O Frame**
   - H2O Frame

5. **H2O**
   - h2o_df object created in R

   - Return pointer to data in REST API JSON Response

6. **H2O Frame**
   - H2O Frame

7. **H2O**
   - H2O Frame

8. **H2O**
   - H2O Frame

9. **H2O**
   - H2O Frame

10. **H2O**
    - H2O Frame

**3.1 HDFS provides data**

**3.2 Distributed H2O Frame in DKV**

**3.3 Return pointer to data in REST API JSON Response**

**3.4 h2o_df object created in R**
R Script Starting H2O GLM

```
R script

.h2o.startModelJob()
POST /3/ModelBuilders/glm

.h2o.glm()

http
REST/JSON

TCP/IP

HTTP

Network layer

REST layer

H2O - algos

H2O - core

User process

H2O process

Legend

R script

Standard R process

H2O process

Job

GLM algorithm

GLM tasks

Fork/Join framework

K/V store framework

H2O process

H2O - algos

H2O - core

User process

H2O process

Legend

Network layer

REST layer

H2O - algos

H2O - core

User process

H2O process

R script

Standard R process

H2O process

TCP/IP

HTTP

REST/JSON

.h2o.startModelJob()
POST /3/ModelBuilders/glm

.h2o.glm()
R Script Retrieving H2O GLM Result

HTTP
REST/JSON
h2o.getModel()
GET /3/Models(glm_model_id
h2o.glm()
R script
Standard R process

TCP/IP
HTTP
REST/JSON
h2o.get_model() GET /3/Models/glm_model_id
h2o.glm()
R script
Standard R process

Legend
Network layer
REST layer
H2O - algos
H2O - core
User process
H2O process

Fork/Join framework
K/V store framework
H2O process
H2O Demo!
Thank You