Machine Learning: Machine Learning tools

Unige

Kevin Raymundo Serrano Vilchis

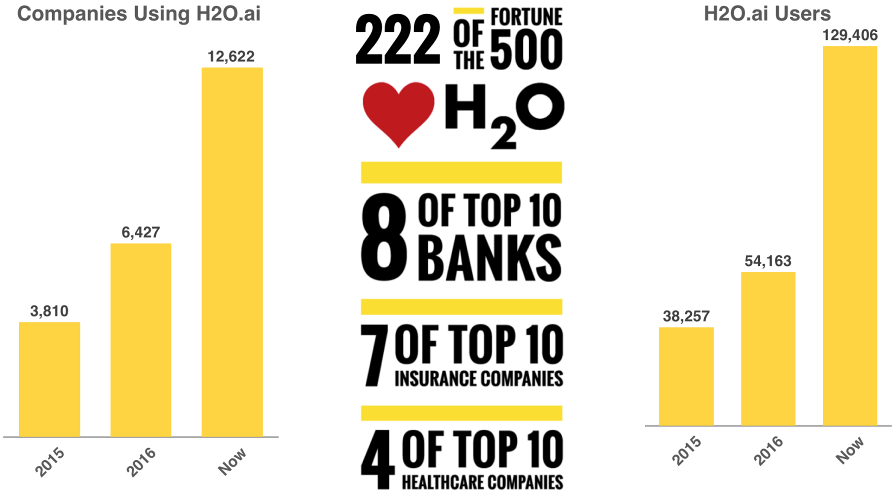
S4539351

H_2O

Overview

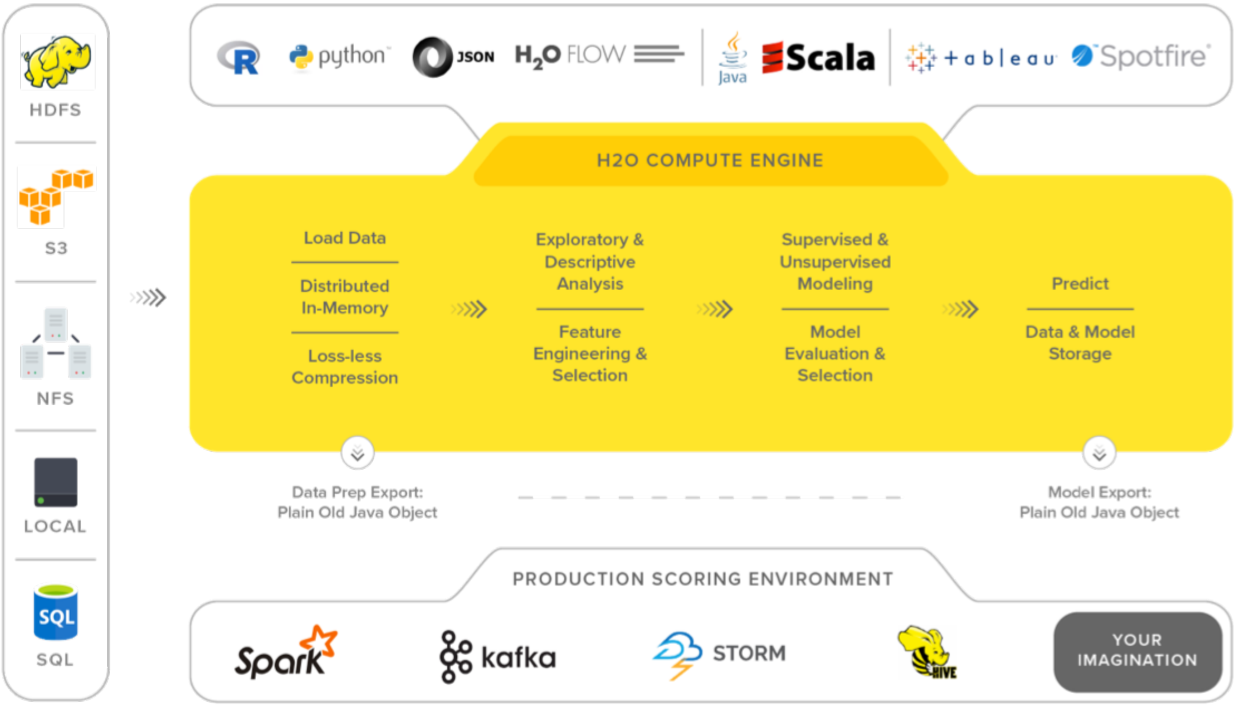
- Introduction H2O
- Architechture
- Setup H2O
- FLOW API (live 'coding')
- H2O using Python (live coding)
- Summary

Introduction



- Founded in 2011 in Silicon Valley (formerly 0xdata)
- #1 Open-source machine learning platform for enterprises
- The company receives fees for providing customer service and customized extensions.
- Platform:
  - **Parallelized and distributed algorithms** to make the most out of **multithreaded** systems.
  - Easy to use and adopt
  - **Big data** + Better models = Better predictions
- Comcast, Macy's, Cisco, PayPal

Architecture


HDFS
S3
NFS
LOCAL
SQL
R
python
JSON
H2O FLOW
Java
Scala
+ a b | e a u
Spotfire
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

- Distributed file systems + stream processing platforms + APIs
- Data stays on DFS, on the API side we get a pointer to the distributed dataset.
- Also possible to actually import it into workspace using data frames.
- They can also interface between other packages like caffe, tensorflow, etc...

Setup

Prerequisites to launch H2O and Flow

- 64 bit Java 6+

Flow users

1. Download and unpack h2o zip file from website link (http://h2o-release.s3.amazonaws.com/h2o/rel-wheeler/4/index.html)
2. Run the following command from terminal

```
cd ~/Downloads
unzip h2o-3.16.0.4.zip
cd h2o-3.16.0.4
java -jar h2o.jar
```

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

Python users

1. Prerequisite: Python installed (versions 2.7.x, 3.5.x, 3.6.x)
2. Using pip, install dependencies and h2o

```
pip install requests
pip install tabulate
pip install scikit-learn
pip install colorama
pip install future
pip install h2o
```

3. Check that library is properly installed:

```
import h2o
h2o.init(nthreads = -1)
```

R users

1. Prerequisite: R installed (version 3 or later)
2. Install from cran

```
# Download packages that H2O depends on.
pkgs <- c("RCurl","jsonlite")
for (pkg in pkgs) {
if (! (pkg %in% rownames(installed.packages()))) { install.packages(pkg) }
}
# Download, install and initialize the H2O package for R.
install.packages("h2o", type="source", repos="http://h2o-release.s3.amazonaws.com/h2o/rel-wheeler/4/R")
```

3. Check that library is properly installed:

```
library(h2o)
h2o.init(nthreads = -1)
```

In [ ]:
```
library(h2o)
h2o.init(nthreads = -1)
```

Small demo with MNIST dataset

In [ ]:
```
# This step takes a few seconds bc we have to download the data from the
internet...
train_file <- "https://h2o-public-test-data.s3.amazonaws.com/bigdata/lap
top/mnist/train.csv.gz"
test_file <- "https://h2o-public-test-data.s3.amazonaws.com/bigdata/lapt
op/mnist/test.csv.gz"
train <- h2o.importFile(train_file)
test <- h2o.importFile(test_file)
```

In [ ]:
```
y <- "C785"                    # response column: digits 0-9
x <- setdiff(names(train), y)  # vector of predictor column names
```

In [ ]:
```
# Since the response is encoded as integers, we need to tell H2O that
# the response is in fact a categorical/factor column.  Otherwise, it
# will train a regression model instead of multiclass classification.
train[,y] <- as.factor(train[,y])
test[,y] <- as.factor(test[,y])
```

In [ ]:
```
dl_fit1 <- h2o.deeplearning(x = x,
                            y = y,
                            training_frame = train,
                            model_id = "dl_fit1",
                            hidden = c(20,20),
                            seed = 1)
```

In [ ]:

```
dl_fit3 <- h2o.deeplearning(x = x,
                            y = y,
                            training_frame = train,
                            validation_frame = test,
                            model_id = "dl_fit3",
                            epochs = 50,
                            sparse = TRUE,
                            hidden = c(128,64),
                            activation = "RectifierWithDropout",
                            input_dropout_ratio = 0.2,
                            hidden_dropout_ratios = c(0.3, 0.2),
                            # nfolds = 0,                            #us
ed for early stopping
                            score_interval = 1,                    #used
for early stopping
                            stopping_rounds = 5,                   #used
for early stopping
                            stopping_metric = "misclassification", #used
for early stopping
                            stopping_tolerance = 1e-3,             #used
for early stopping
                            seed = 1)
```

In [ ]:

```
h2o.scoreHistory(dl_fit3)
```

In [ ]:

```
h2o.confusionMatrix(dl_fit3)
```

In [ ]:

```
plot(dl_fit3,
     timestep = "epochs",
     metric = "classification_error")
```

Sumary

- H2O is easy to use
- Off-the-shelf algorithms
- FLOW API is targeted to users who prefer GUIs or have basic coding experience
- Extern libraries/packages can be added by using Python and R
    - Data analysis and pre-processing

In [ ]:

```
h2o.shutdown()
```