Distributed GLM

Tom Kraljevic

January 27, 2015

Atlanta, GA @ Polygon

Outline for today's talk

• About H2O.ai (the company)	(5 minutes)
• About H2O (the software)	(10 minutes)
• H2O's Distributed GLM	(30 minutes)
• Demo of GLM	(15 minutes)
• Q & A	(up to 30 minutes)

Content for today's talk can be found at:

https://github.com/h2oai/h2o-meetups/tree/master/2015_01_27_GLM

H2O.ai Overview

- Founded: 2011 venture-backed, debuted in 2012
- Product: H2O open source in-memory prediction engine
- Team: 30
- HQ: Mountain View, CA
- SriSatish Ambati – CEO & Co-founder (Founder Platfora, DataStax; Azul)
- Cliff Click – CTO & Co-founder (Creator Hotspot, Azul, Sun, Motorola, HP)
- Tom Kraljevic – VP of Engineering (CTO & Founder Luminix, Azul, Chromatic)


Distributed

Systems

Engineers

Making

ML Scale!

H_2O.ai
Machine Intelligence

Scientific Advisory Council

Stephen Boyd
Professor of EE Engineering
Stanford University

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

Trevor Hastie
Professor of Statistics
Stanford University

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, 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

Algorithms on H_2O

Supervised Learning

Statistical Analysis

- **Generalized Linear Models**: Binomial, Gaussian, Gamma, Poisson and Tweedie
- **Cox Proportional Hazards Models**
- **Naïve Bayes**

Ensembles

- **Distributed Random Forest**: Classification or regression models
- **Gradient Boosting Machine**: Produces an ensemble of decision trees with increasing refined approximations

Deep Neural Networks

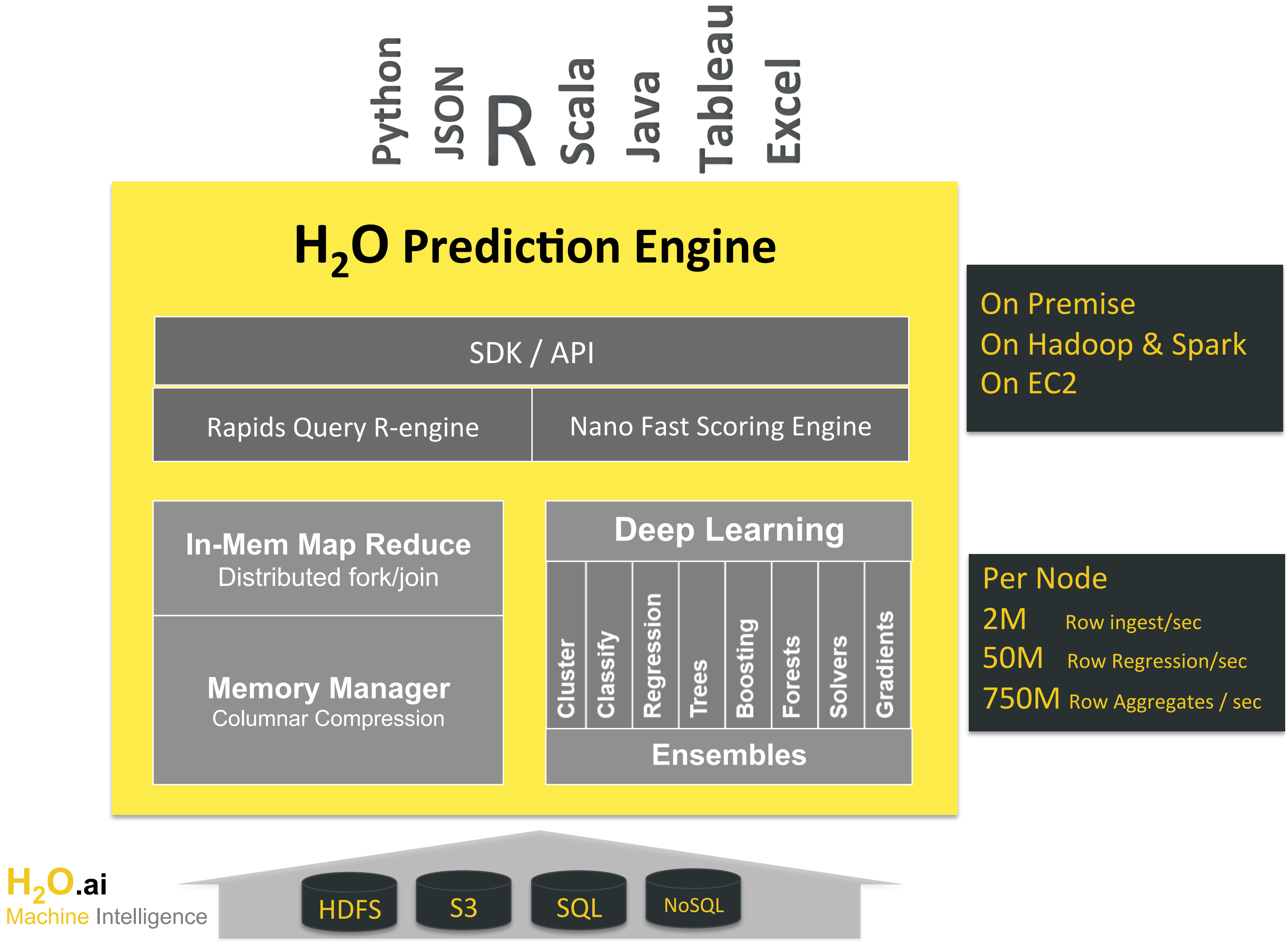
- **Deep learning**: Create multi-layer feed forward neural networks starting with an input layer followed by multiple layers of nonlinear transformations

Algorithms on H_2O

Unsupervised Learning

Clustering	• **K-means**: Partitions observations into k clusters/groups of the same spatial size
Dimensionality Reduction	• **Principal Component Analysis**: Linearly transforms correlated variables to independent components
Anomaly Detection	• **Autoencoders**: Find outliers using a nonlinear dimensionality reduction using deep learning

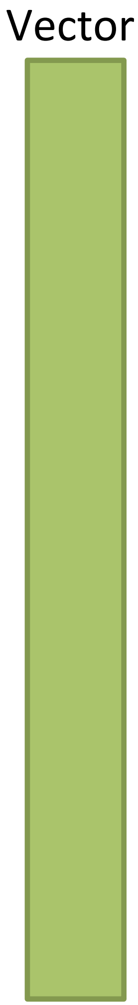
H_2O.ai
Machine Intelligence


Python
JSON
R
Scala
Java
Tableau
Excel
H2O Prediction Engine
SDK / API
Rapids Query R-engine
Nano Fast Scoring Engine
In-Mem Map Reduce
Distributed fork/join
Memory Manager
Columnar Compression
Deep Learning
Cluster
Classify
Regression
Trees
Boosting
Forests
Solvers
Gradients
Ensembles
On Premise
On Hadoop & Spark
On EC2
Per Node
2M Row ingest/sec
50M Row Regression/sec
750M Row Aggregates / sec
HDFS
S3
SQL
NoSQL
H2O.ai
Machine Intelligence

Distributed GLM

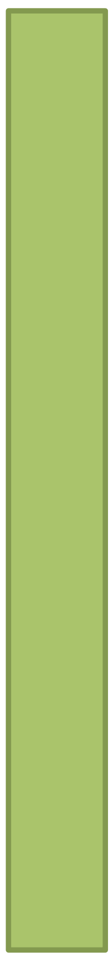
H2O.ai
Machine Intelligence

Distributed Data Taxonomy



Distributed Data Taxonomy

Vector



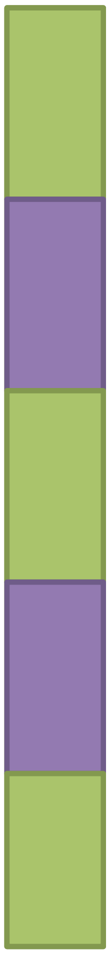
The vector may be very large (billions of rows)

- Stored as a compressed column (often 4x)
- Access as Java primitives with on-the-fly decompression
- Support fast Random access
- Modifiable with Java memory semantics

H2O.ai
Machine Intelligence

Distributed Data Taxonomy

Vector

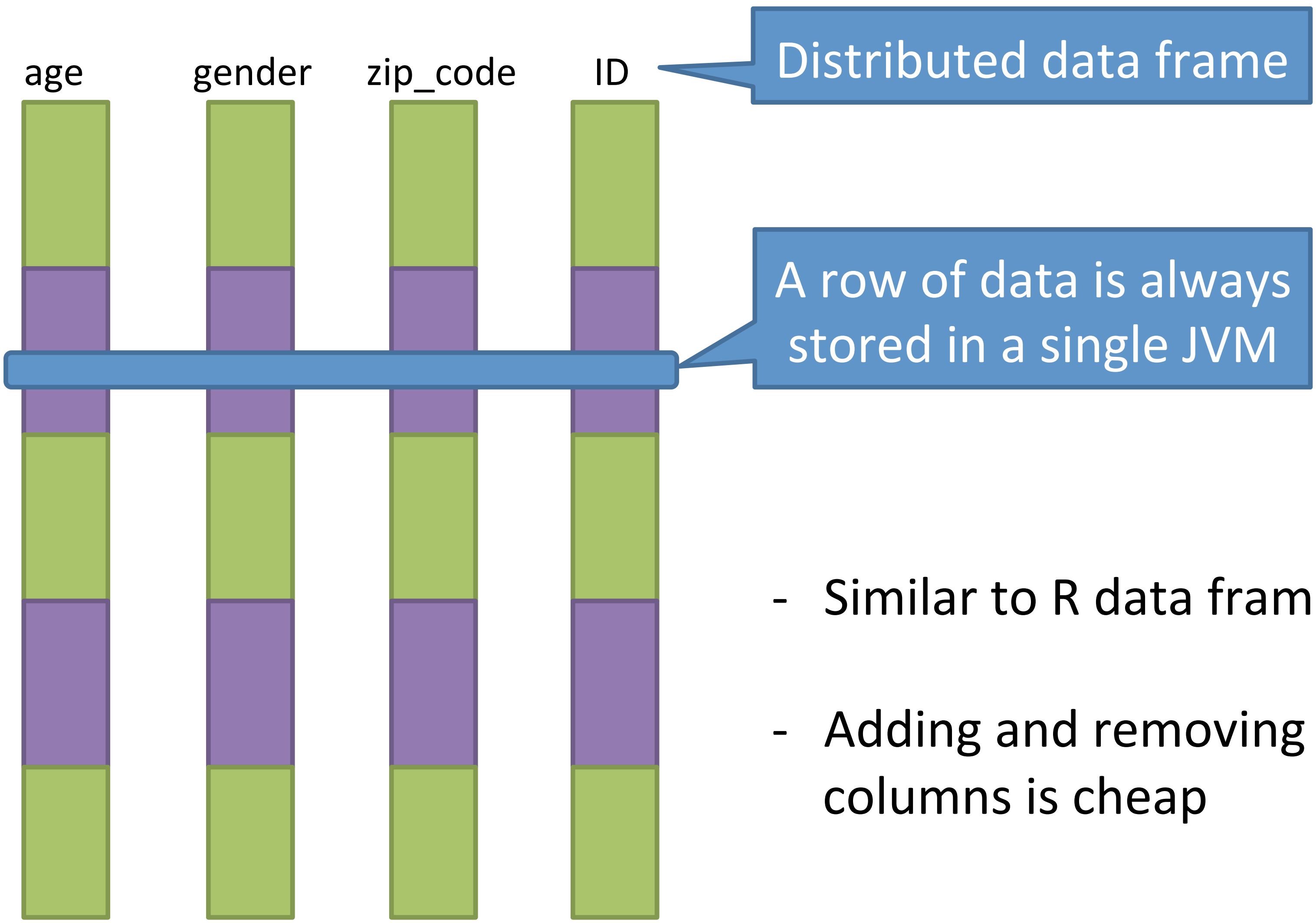


Large vectors must be distributed over multiple JVMs

- Vector is split into chunks
- Chunk is a unit of parallel access
- Each chunk ~ 1000 elements
- Per-chunk compression
- Homed to a single node
- Can be spilled to disk
- GC very cheap

H_2O.ai
Machine Intelligence

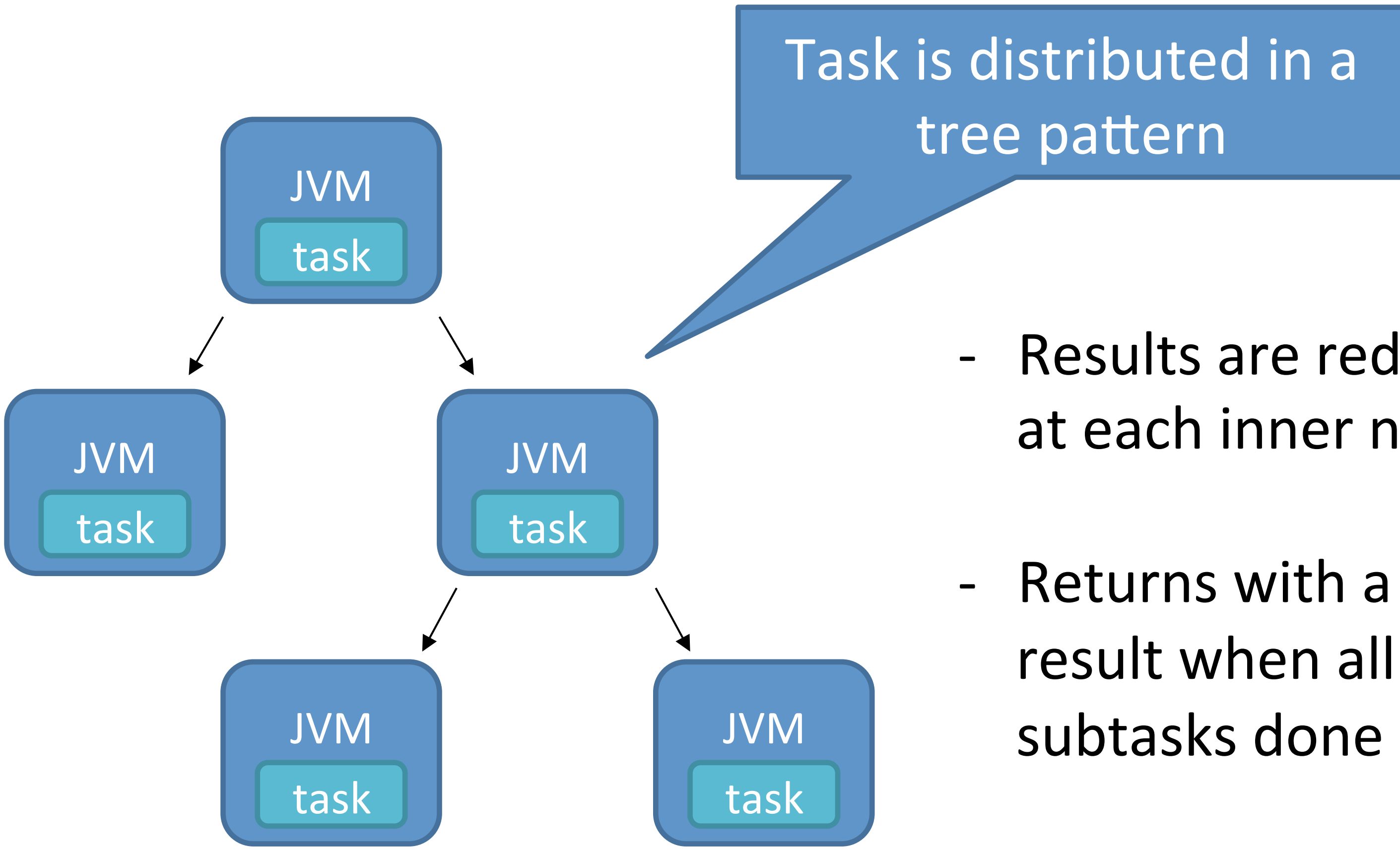
Distributed Data Taxonomy



- Similar to R data frame
- Adding and removing columns is cheap

H2O.ai Machine Intelligence

Distributed Fork/Join

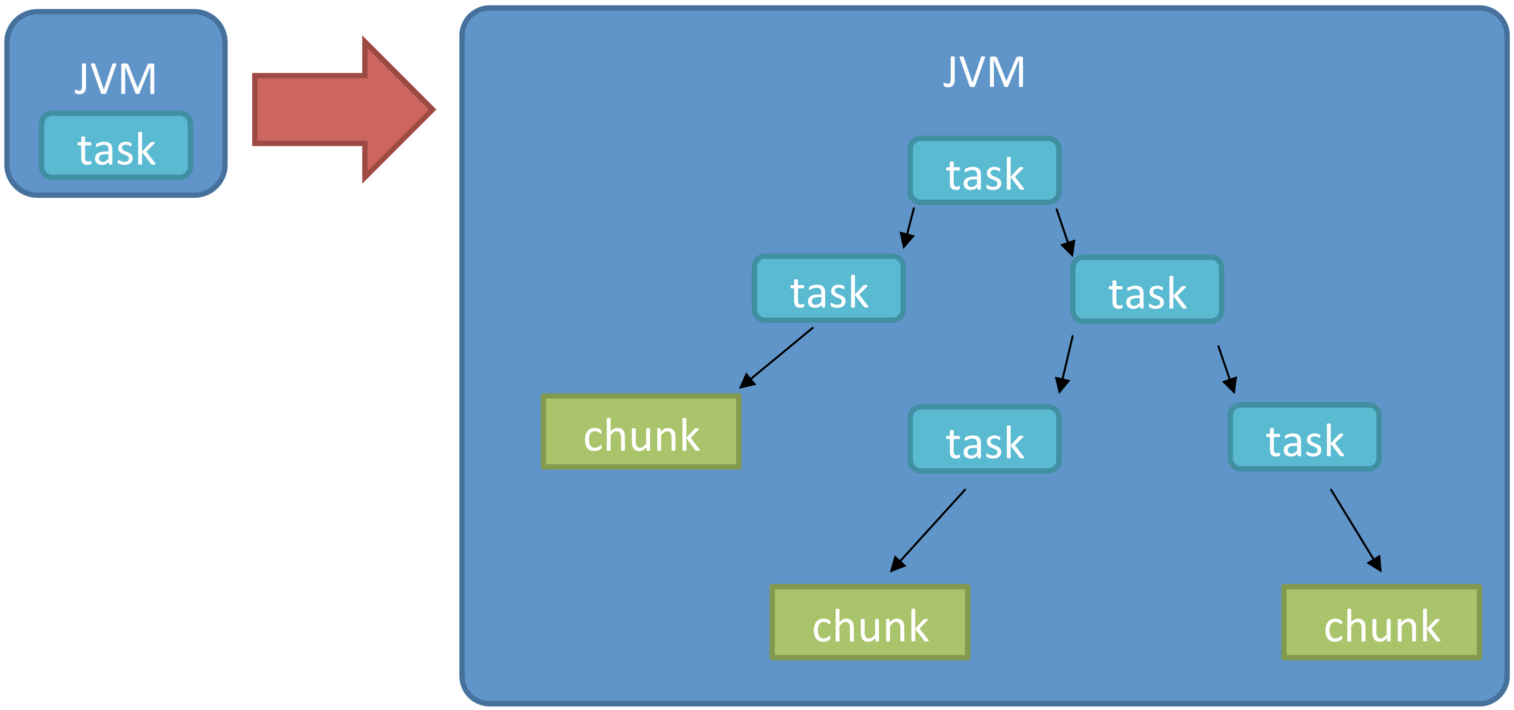


- Results are reduced at each inner node

- Returns with a single result when all subtasks done

H2O.ai
Machine Intelligence

Distributed Fork/Join



On each node the task is parallelized using Fork/Join

H2O's GLM Fit

$$\min_{\beta}\left(\frac{1}{N}\text{log\_likelihood}(\text{family}, \beta) + \lambda\left(\alpha\|\beta\|_1 + \frac{1-\alpha}{2}\|\beta\|_2\right)\right)$$

Classic GLM

Regularization penalty

$$E(y) = \text{link}^{-1}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_n x_n)$$

Model interpretability

H2O.ai
Machine Intelligence

Features of H2O's GLM

- Support for various GLM families
  - Gaussian (numeric regression)
  - Binomial (binary classification / logistic regression)
  - Poisson
  - Gamma
- Regularization
  - L1 (Lasso) (+ Strong rules)
  - L2 (Ridge regression)
  - Elastic-net
- Automatic and efficient handling of categorical variables
- Efficient distributed n-fold cross validation
- Grid search over elastic-net parameter α
- Lambda search for best model
- Upper and lower bounds for coefficients

Input Parameters: Predictors, Response and Family

```
df = iris
df$is_setosa = df$Species == 'setosa'

library(h2o)
h = h2o.init()
h2odf = as.h2o(h, df)

# Y is a T/F value
y = 'is_setosa'
x = c("Sepal.Length", "Sepal.Width",
      "Petal.Length", "Petal.Width")
binary_classification_model =
  h2o.glm(data = h2odf, y = y, x = x, family = "binomial")

# Y is a real value
y = 'Sepal.Length'
x = c("Sepal.Width",
      "Petal.Length", "Petal.Width")
numeric_regression_model =
  h2o.glm(data = h2odf, y = y, x = x, family = "gaussian")
```

H2O.ai
Machine Intelligence

Input Parameters: Number of iterations

```
# Specify maximum number of IRLSM iterations
# (default is 100)

h2o.glm(…, iter.max = 50)
```

Input Parameters: Regularization

```
# Enable lambda_search
h2o.glm(…, lambda_search = TRUE)

# Enable strong_rules (requires lambda_search)
h2o.glm(…, lambda_search = TRUE, strong_rules = TRUE)

# Specify max_predictors (requires lambda_search)
h2o.glm(…, lambda_search = TRUE, max_predictors = 100)

# Grid search over alpha
h2o.glm(…, alpha = c(0.05, 0.5, 0.95))

# Turn off regularization entirely
h2o.glm(…, lambda = 0)
```

Input Parameters: Cross validation

```
# Specify 5-fold cross validation

model_with_5folds = h2o.glm(data = h2odf, y = y, x = x,
family = "binomial", nfolds = 5)

print(model_with_5folds@model$auc)
print(model_with_5folds@xval[[1]]@model$auc)
print(model_with_5folds@xval[[2]]@model$auc)
print(model_with_5folds@xval[[3]]@model$auc)
print(model_with_5folds@xval[[4]]@model$auc)
print(model_with_5folds@xval[[5]]@model$auc)
```

Outputs

- Coefficients
- Normalized coefficients (variable importance)

```
Coefficients:
        Dest.ABQ    Dest.ACY     Dest.ALB    Dest.AMA    Dest.ANC
         0.80322    -0.06288      0.13333     0.14092     0.92581
        Dest.AT     Dest.AUS     Dest.AVL    Dest.AVP    Dest.BDL
        -0.21849     0.78392     -0.34974    -0.31825     0.38924

                                      .   .
                                      .   .
                                      .   .

        DayofMonth   DayOfWeek   CRSDepTime CRSArrTime FlightNum
        -0.03087      0.02110     0.00029     0.00027     0.00007
        Distance     Intercept
         0.00024      136.69614
```

H2O.ai
Machine Intelligence

Outputs

- Null deviance, residual deviance
- AIC (Akaike information criterion)
- Deviance explained

```
Degrees of Freedom: 43942 Total (i.e. Null);  43668 Residual
Null Deviance:     60808
Residual Deviance: 54283  AIC: 54833
Deviance Explained: 0.10731
```

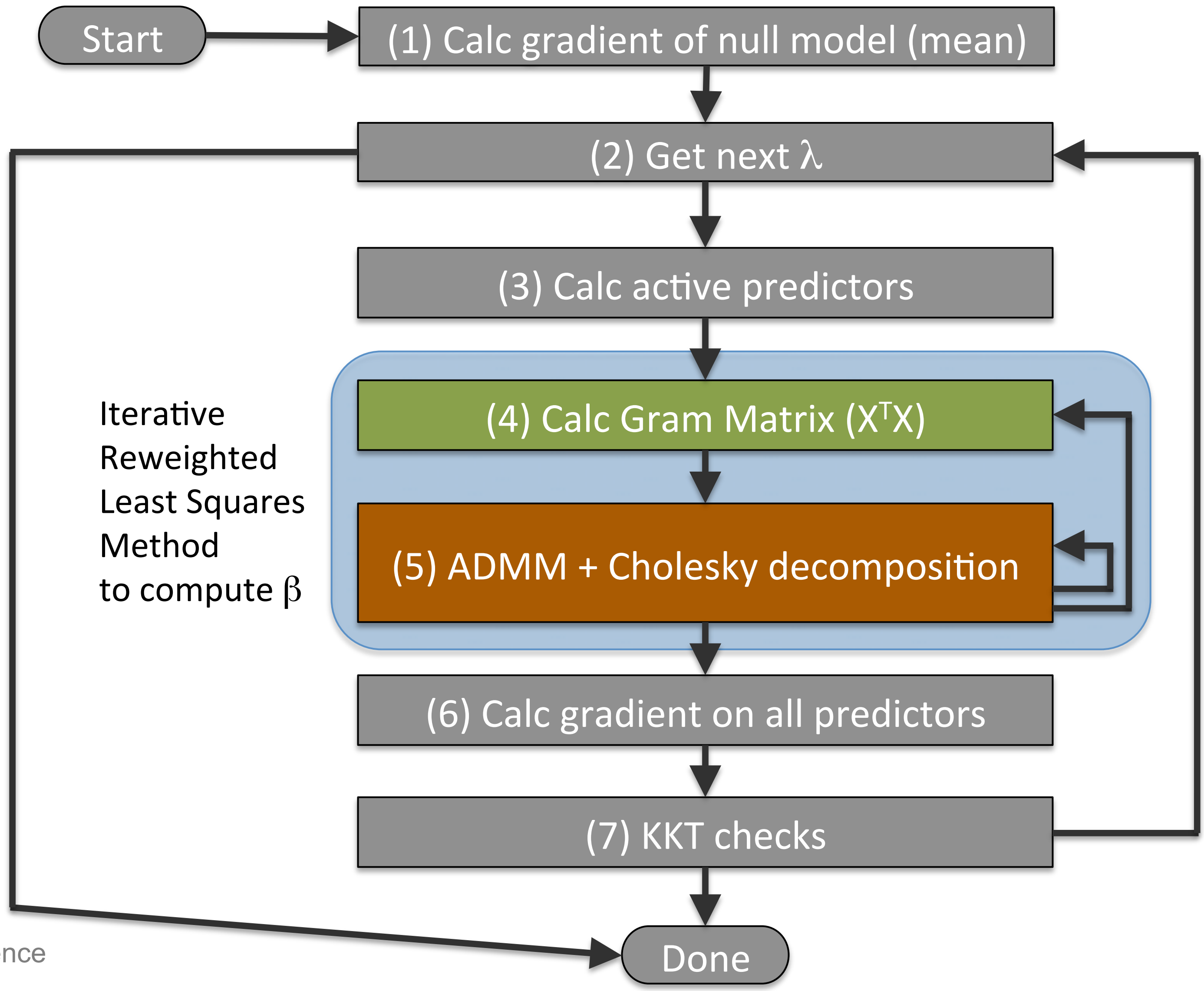
Outputs

- Confusion matrix (for logistic regression)
- AUC (for logistic regression)

```
Confusion Matrix:
        Predicted
Actual   false  true   Error
  false   6747 14126 0.67676
  true    2490 20580 0.10793
  Totals  9237 34706 0.37813

AUC =  0.7166454 (on train)
```

GLM Lifecycle



H2O.ai
Machine Intelligence

GLM Runtime Cost

	CPU	Memory
Calc Gram Matrix (X^TX)	$O(\frac{M * N^2}{p * n})$	$\boxed{O(M * N)} + O(N^2 * p * n)$ (the training data)
ADMM + Cholesky decomposition	$O(\frac{N^3}{p})$	$O(N^2)$

M	Number of rows in the training data
N	Number of predictors in the training data
p	Number of CPUs per node
n	Number of nodes in the cluster

H2O.ai
Machine Intelligence

Best Practices

- GLM works best on tall and skinny datasets
  - But if you have a wide dataset, use L1 penalty and Strong Rules to eliminate columns from the model
- Give lambda search a shot
  - But specify *strong_rules* and/or *max_predictors* if it is taking too long
  - 90% of the time is spent on the larger models with the small lambdas, so specifying *max_predictors* helps a lot
- Keep a little bit of L2 for numerical stability (i.e. don't use alpha 1.0, use 0.95 instead)
- Use symmetric nodes in your cluster
- Bigger nodes can help the ADMM / Cholesky run faster
- Impute if you need to before running GLM

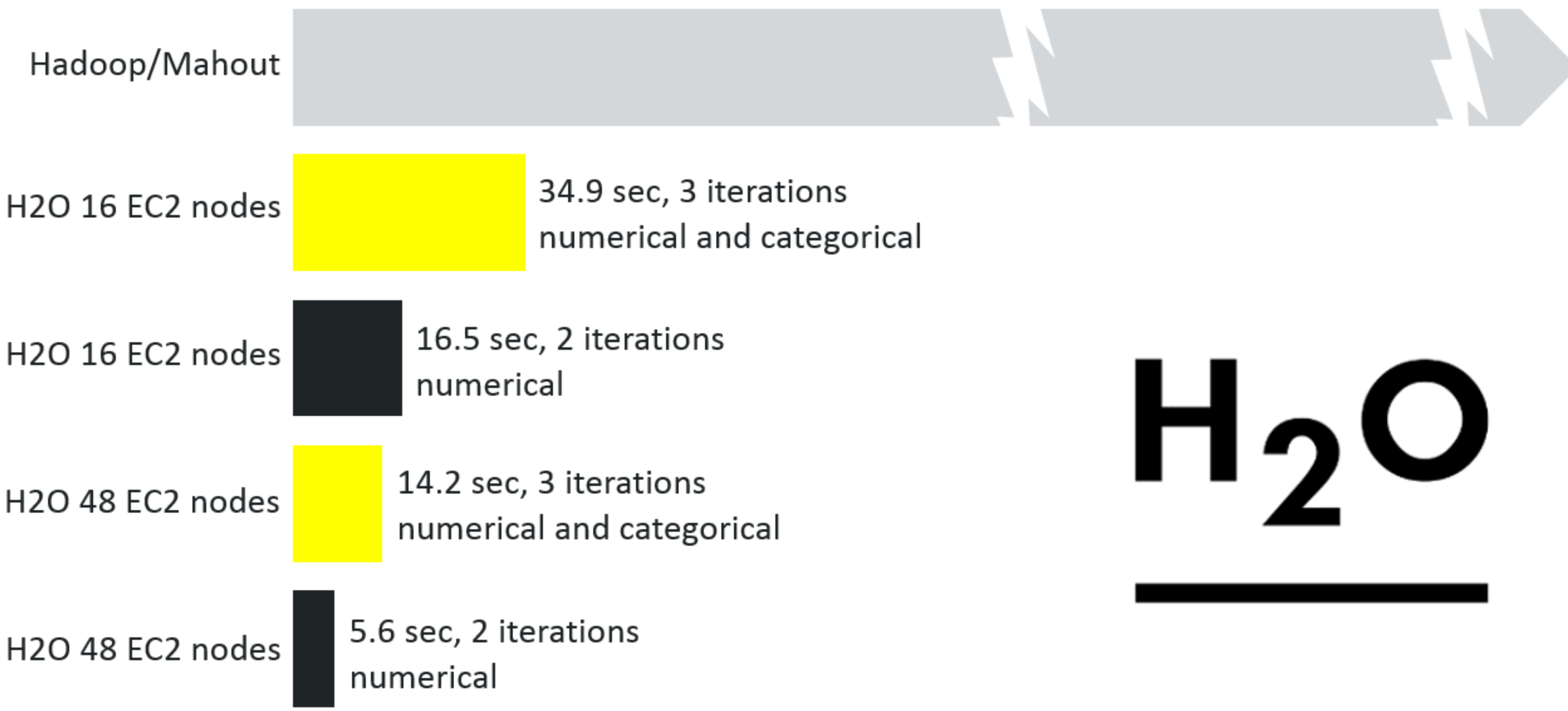
H2O.ai
Machine Intelligence

Things to Watch Out for

- Look for suspiciously different cross-validation results between folds
- Look for explained deviance
  - Too close to 0: model doesn't predict well
  - Too close to 1: model predicts "too" well (one of your input cols is cheating)
- Same for AUC
  - Too close to 0.5: model doesn't predict well
  - Too close to 1: model predicts "too" well
- See if GLM stops early for a particular lambda that interests you (performing all the iterations probably means the solution isn't good)
- Too many N/As in your data (GLM discards rows with N/A values)
  - If you have a really bad column, you might accidentally be losing all your rows.

H2O Billion Row Machine Learning Benchmark

GLM Logistic Regression



Compute Hardware: AWS EC2 c3.2xlarge - 8 cores and 15 GB per node, 1 GbE interconnect

Airline Dataset 1987-2013, 42 GB CSV, 1 billion rows, 12 input columns, 1 outcome column
9 numerical features, 3 categorical features with cardinalities 30, 376 and 380

H2O.ai
Machine Intelligence

Demonstration

Q & A

Thanks for attending!

Content for today's talk can be found at:

https://github.com/h2oai/h2o-meetups/tree/master/2015_01_27_GLM

H2O.ai
Machine Intelligence