

MACHINE LEARNING COM H2O E SPARK

Big Data

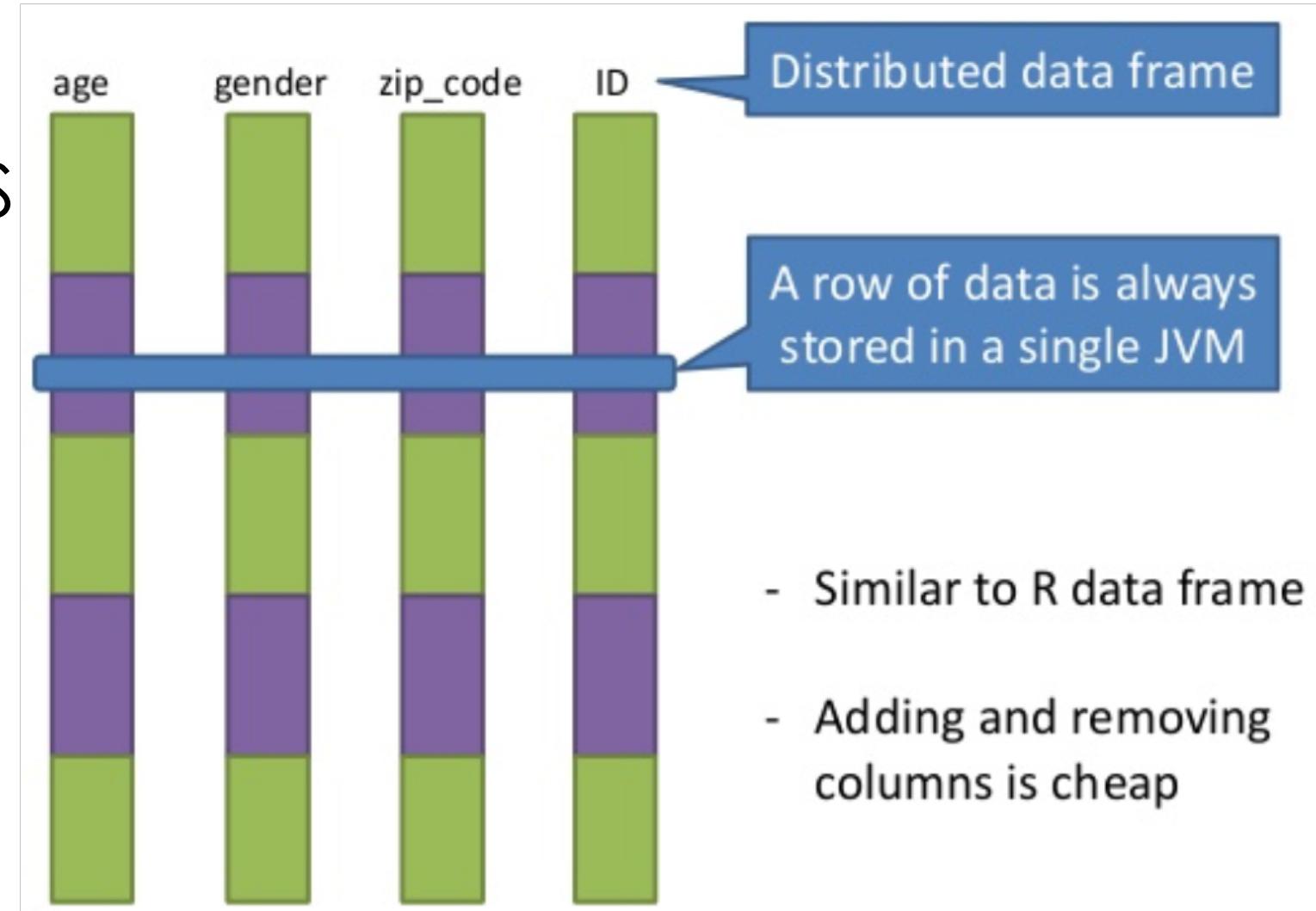
H2O

- Plataforma Open Source de análise preditiva e de Machine Learning
- Distribuído e escalável
- Fornece recursos para construir modelos de ML em Big Data com recursos para publicação em ambientes corporativos.
- Desenvolvido principalmente em Java.

ORGANIZAÇÃO DOS DADOS

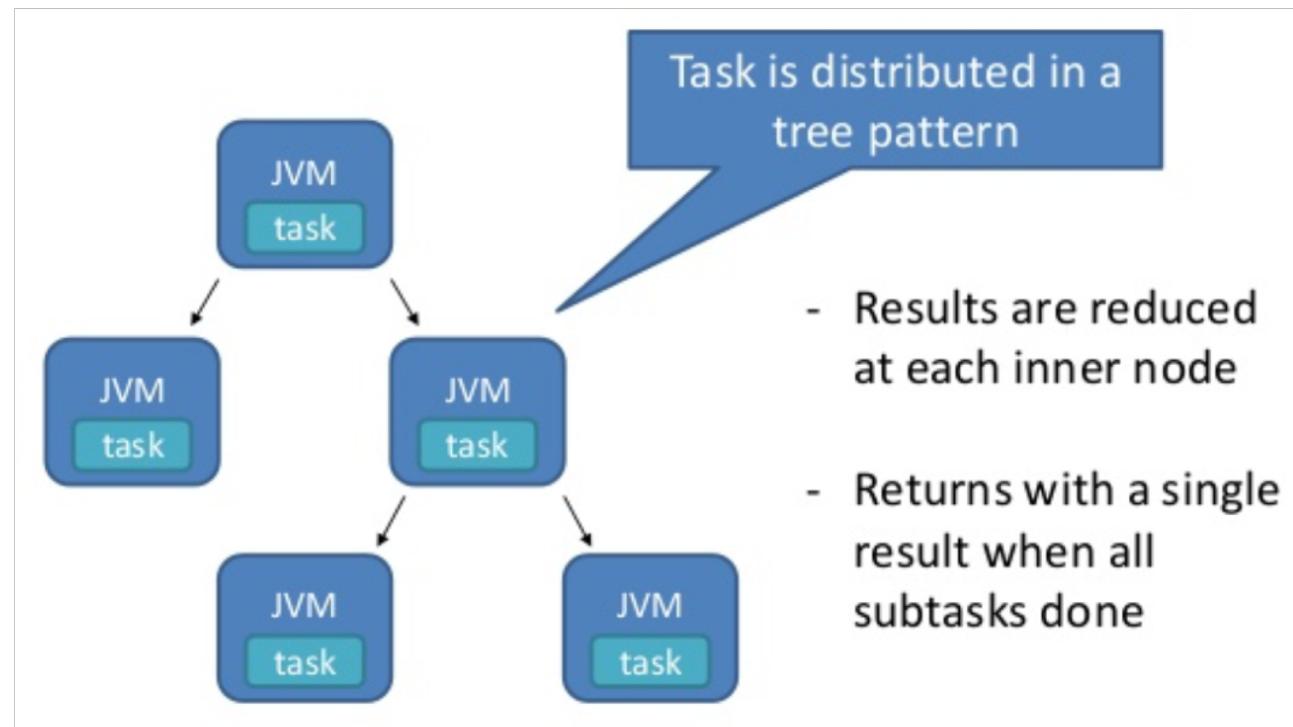
- Sua principal estrutura é um Key-Value Store Distribuído (DKV), usado para acessar e referenciar dados, modelos, objetos, etc., em todos os nós / máquinas
- Dados armazenados como vetores (modelo colunar)
 - Vetores divididos em *chunks*, de aproximadamente 1000 elementos cada
 - O *chunk* é a unidade de processamento paralelo (similar à partição)
 - Chunks distribuídos, podendo estar em memória ou disco
 - Compressão por *chunk*
- Diferente do Spark, os dados são mutáveis

DATA FRAMES, VECTORS & CHUNKS

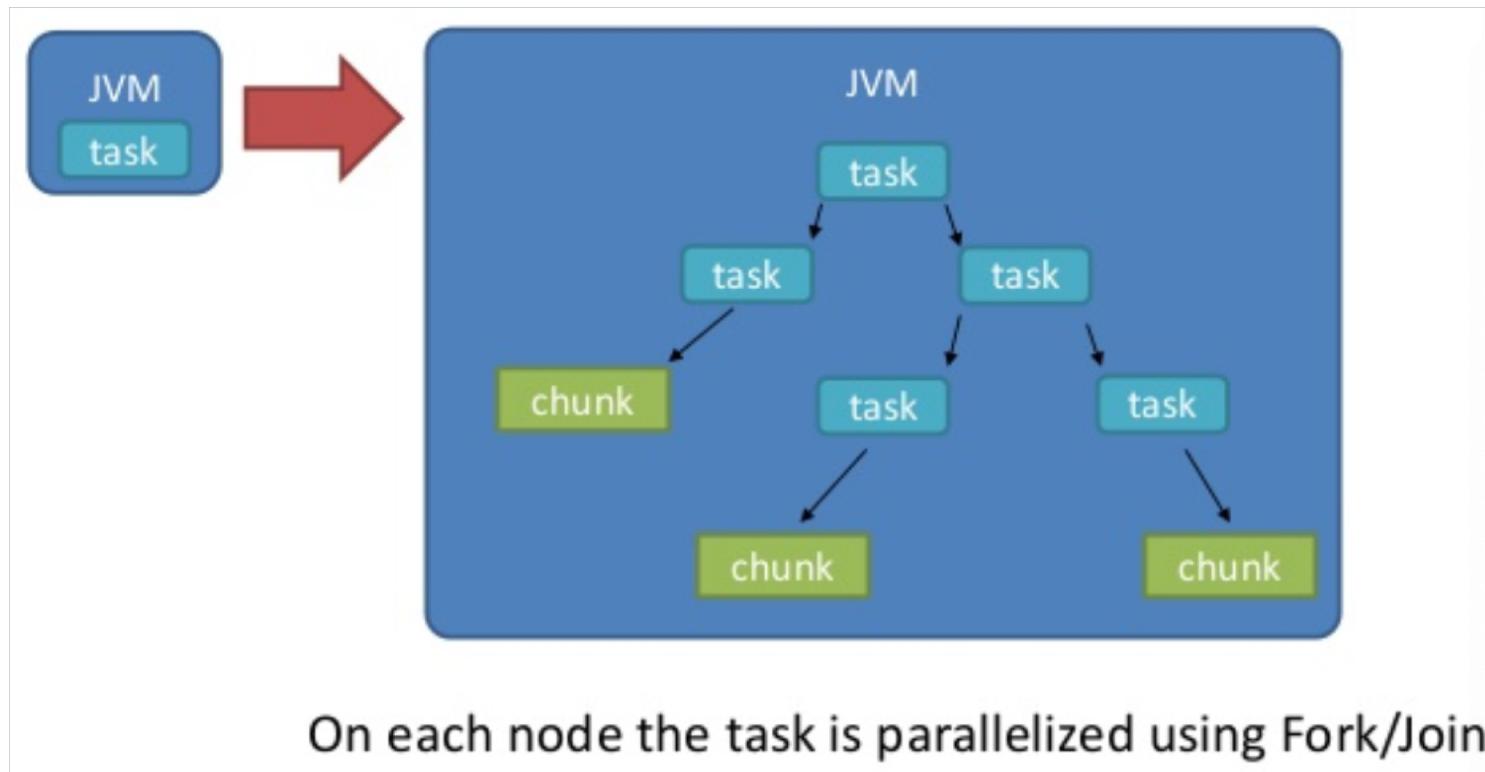


MODELO DE PROCESSAMENTO

- Algoritmos são implementados no paradigma Map Reduce utilizando o framework Java Fork / Join.

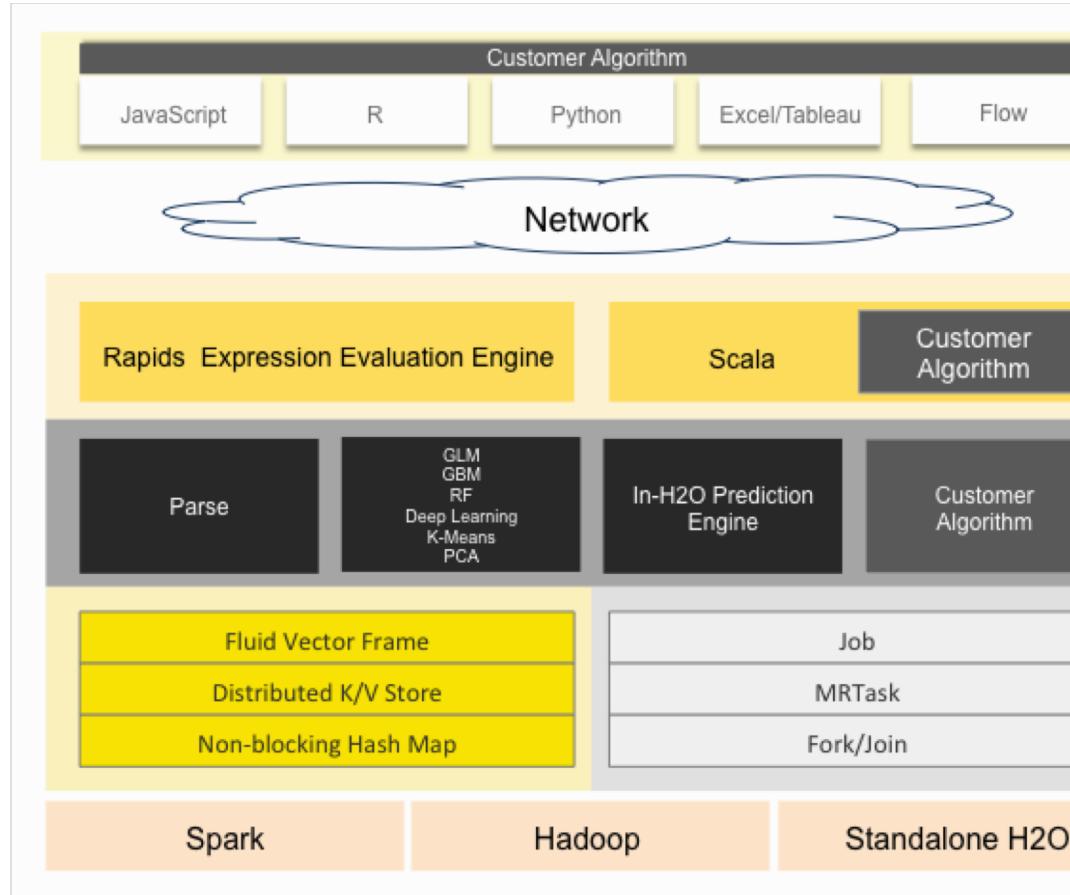


MODELO DE PROCESSAMENTO



PROGRAMAÇÃO

- Os dados são lidos em paralelo e são distribuídos pelo cluster, armazenados na memória em um formato colunar de forma compacta.
- Possui uma API REST para acesso a todos os recursos a partir de um programa ou script externo, via JSON sobre HTTP.
 - A API é usada pela interface Web do H2O (Flow UI), pelo R (H2O-R) e pelo Python (H2O-Python).



ARQUITETURA

ALGORITMOS

- Disponibiliza algoritmos para Deep Learning, Gradient Boosted Trees, Random Forest, Decision Trees, GLM, K-means, PCA, etc

SPARKLING WATER

- Integração do H2O no ecossistema Spark, facilita o uso do H2O em workflows do Spark.
- Projetado como um aplicativo Spark comum
 - fornece uma maneira de iniciar os serviços do H2O em cada nó do cluster Spark e acessar dados armazenados nas estruturas de dados do Spark e do H2O.

SPARKLING WATER

- O Diver cria um SparkContext (sc) que, por sua vez, é usado para criar um H2OContext (hc), que é usado para iniciar os serviços de H2O nos Executors.
- H2OContext é uma conexão ao cluster de H2O e também facilita a comunicação entre o H2O e o Spark.
- Quando um H2OCluster é iniciado, ele possui a mesma topologia que o cluster do Spark e os nós de H2O compartilham as mesmas JVMs que os Executores do Spark.

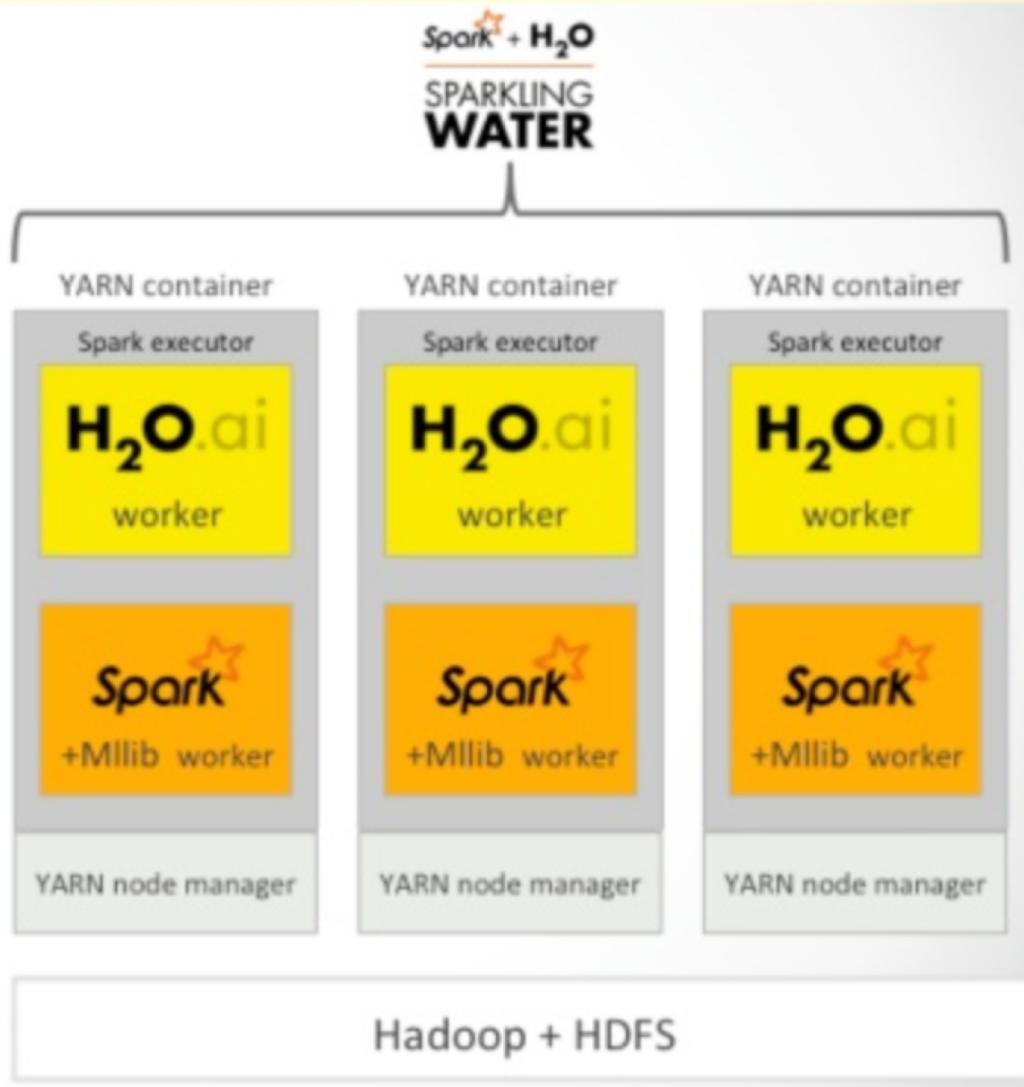
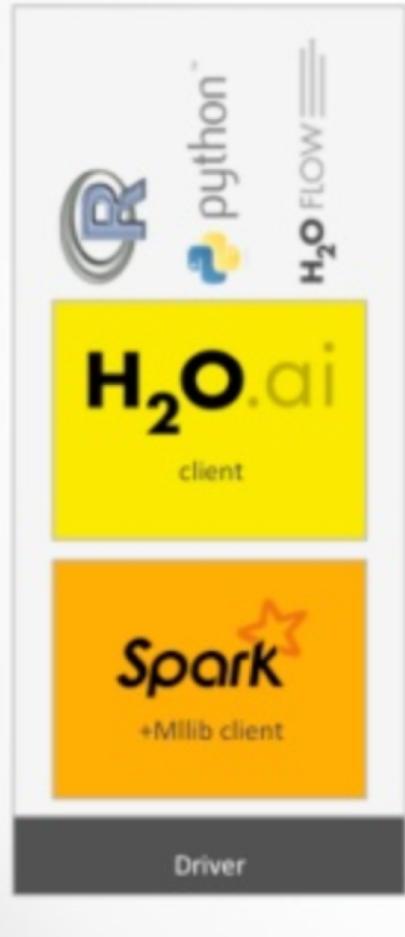
SPARKLING WATER

- Os dados no cluster do Spark, armazenados como um RDD, precisam ser convertidos em um H2OFrame (Dataframes distribuídos do H2O).
 - Isso requer uma cópia de dados devido à diferença no layout de dados no Spark e no H2O.
 - Sobrecarga é baixa devido à compressão de dados do H2O.

SPARKLING WATER

- Ao converter um H2OFrame em RDD, o Sparkling Water cria um wrapper em torno do H2OFrame para fornecer uma API semelhante à RDD.
 - Nesse caso, nenhum dado é duplicado e os dados são exibidos diretamente do H2OFrame subjacente.
 - Como o H2O é executado nas mesmas JVMs que os Executors do Spark, a movimentação de dados do Spark para o H2O ou vice-versa é feita em memória, no mesmo processo Java.

Scala main program



SPARKLING WATER

OPERAÇÕES COM H2OFRAME

- <http://docs.h2o.ai/h2o/latest-stable/h2o-py/docs/frame.html>
- <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-munging.html>
- <http://docs.h2o.ai/h2o/latest-stable/h2o-py/docs/modeling.html>

H2O GLM

- GLM + ElasticNet
- ElasticNet
 - Ridge Regression + LASSO
 - Ridge (L2)
 - Colinearidade, robustez
 - Lasso (L1)
 - Remoção de variáveis
 - Alpha
 - Balanceamento entre Ridge e LASSO
 - Lambda
 - Peso do penalizador
- GLM
 - Diferentes distribuições (não gaussianas)
 - Erro
 - Variância não constante

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

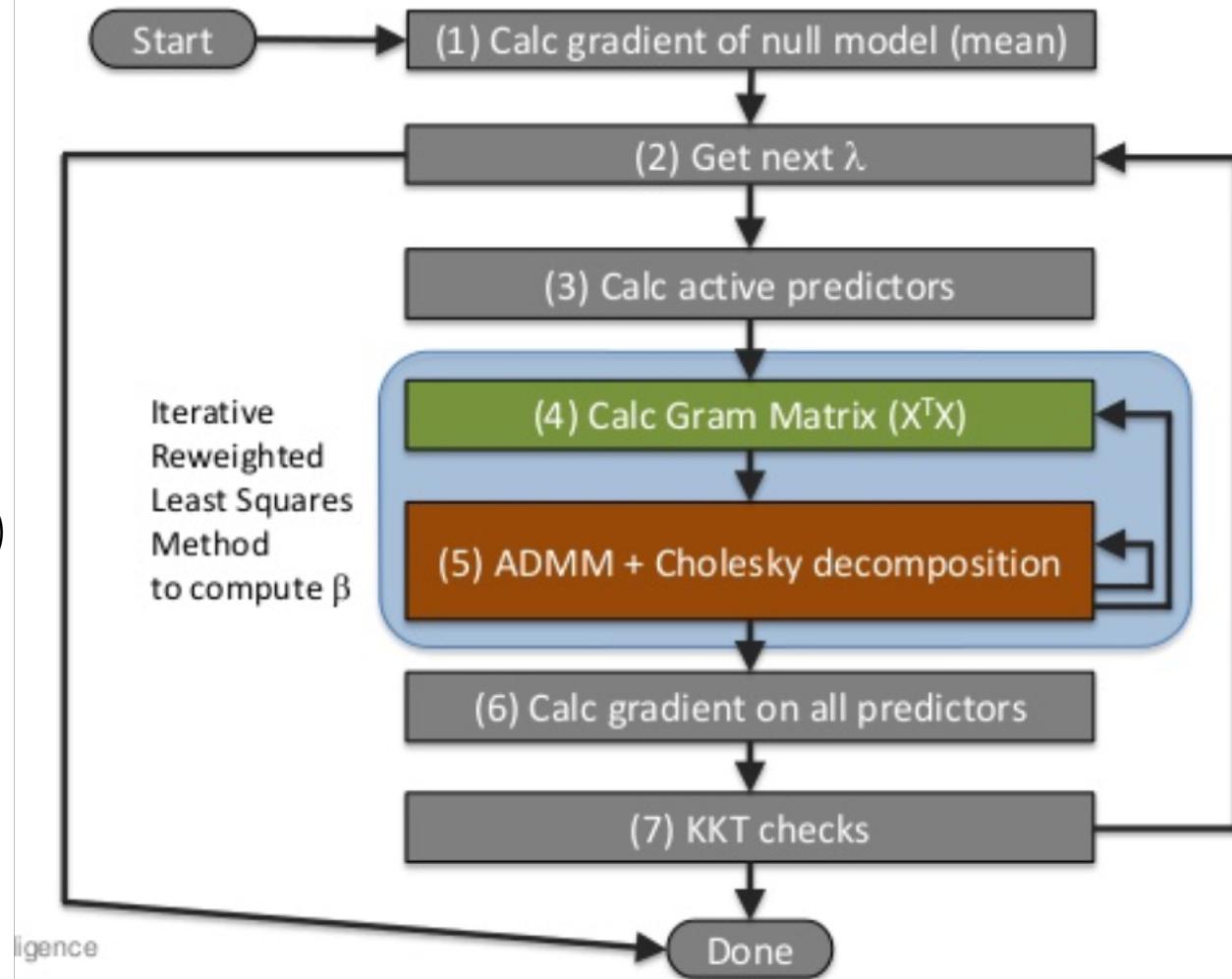
Classic GLM Regularization penalty


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

H2O GLM

- Gram Matrix
 - Matriz cuja Inversa é a solução do OLS
- ADMM + Cholesky
 - Decomposição substituta à inversão de matrizes, com estabilidade numérica
 - adequada a dados esparsos (variáveis categóricas, eliminação de variáveis – Lasso)
- Maiores residuais recebem menor peso (reweight)
 - Peso é aplicado à Gram Matrix ($X^T W X$)
- Repete até convergência
- Detalhes
 - <https://www.h2o.ai/wp-content/uploads/2018/01/GLM-BOOKLET.pdf>

GLM Lifecycle



H2O GLM

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

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



PRÁTICA