Spark ML vs Sparkling Water: An Empirical Analysis of Distributed Machine Learning

Sidharth Singla • Varshanth R Rao

CS848 Course Project
Agenda

1. Motivation
2. Distributed ML Frameworks
3. Overview of ML Algorithms
4. Experimental Setup
5. Research Questions (RQ)
6. RQ Experiments
7. Conclusion
8. Future Work
Motivation

(A) Big Data Distributed Frameworks

B) ML Libraries

(A+B)
# Distributed ML Frameworks

<table>
<thead>
<tr>
<th>Distributed Framework</th>
<th>Spark</th>
<th>Spark</th>
<th>Spark</th>
<th>Spark</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML/DL Support</td>
<td>Strong ML/Basic DL</td>
<td>Strong ML/Strong DL</td>
<td>Strong DL (Keras)</td>
<td>Strong ML/Basic DL</td>
</tr>
<tr>
<td>Data Abstraction</td>
<td>Spark Dataframe</td>
<td>H2O Frame (In mem)</td>
<td>Spark DF/RDD</td>
<td>RDD</td>
</tr>
<tr>
<td>Public Release</td>
<td>Stable</td>
<td>Stable</td>
<td>Stable</td>
<td>Alpha</td>
</tr>
<tr>
<td>Dev Language</td>
<td>Multiple (Scala Base)</td>
<td>Multiple (Scala Base)</td>
<td>Python</td>
<td>Scala</td>
</tr>
</tbody>
</table>
Focus:

(Py) Spark ML vs (Py) Sparkling Water
**ML Algorithms**

**Logistic Regression**
- Solves binary classification problem by assigning probabilities to predictions
- Parameter Learning
- Parameter updates through gradient computation

**K-Means Clustering**
- Unsupervised non parametric learning
- Solves data grouping problem
- Relies on feature similarity of neighbors
ML Algorithms

Principal Component Analysis

- Dimensionality Reduction Technique
- Singular Value Decomposition of Covariance Matrix
- Used to “uplift” the “curse of dimensionality”

Multi-Layer Perceptron (MLP)

- Parametric Universal Function Approximator
- Applies non linearity & parameter updates performed through gradient computation
## ML Algorithms

<table>
<thead>
<tr>
<th></th>
<th>LR</th>
<th>KMC</th>
<th>PCA</th>
<th>MLP</th>
</tr>
</thead>
</table>
| **Time Complexity**
  (Per Iteration)  | $O(n \times d)$ | $O(n \times d \times k)$ | $O(n \times d^2 + d^3)$ | $O(n \times (d \times h_1 + h_1 \times h_2 + \ldots h_m \times c))$ |
| **Space Complexity** | $O(d)$         | $O((n + k) \times d)$ | $O(n \times d + d^2)$ | $O(d \times h_1 + h_1 \times h_2 + \ldots h_m \times c)$ |

$n =$ Number of samples in dataset  
$d =$ Number of features representing each sample  
k =$ Number of clusters  
h$_i =$ Number of hidden nodes at layer $i$  
c =$ Number of output classes
Experimental Setup

Cluster Config

- Spark Standalone Cluster Mode
- 1 Master Node & [1,2,3] Worker Nodes
- 12 Core CPU, 16GB RAM Per Node
- Ganglia monitoring setup on each node with metrics sinks as the driver node

Dataset: Cats vs Dogs

- ~130,000 images of cats and dogs
- 2048-d ResNet 50 features as data
- Total Size = 3GB
- Training:Testing = 5:1
Research Questions

1) Given a dataset of fixed size and fixed cluster:
   a) How does each library compare in terms of runtime & accuracy?
   b) What are the characteristics of the network usage of the libraries?
RQ1a: Dataset Size & Cluster Fixed

Setup Phase Run Time Comparison

- Creating Spark Session
- Assembling Data
- Creating H2O Context
- Creating Spark DataFrame
- Creating H2O Frame

Training Time: pca

Training Time: mlp

Training Accuracy: pca

Testing Accuracy: mlp
RQ1a: Dataset Size & Cluster Fixed
RQ1b: Dataset Size & Cluster Fixed - Network Usage

PCA

MLP

K-Means

Logistic Regression

Sparkling Water
Research Questions

2) **Given a dataset of fixed size:**
   a) Does scale out reduce the runtime linearly as expected?
   b) Does the model accuracy vary with scale out?
RQ2a: Scale Out Effect on Runtime
RQ2b: Scale Out Effect on Accuracy
Research Questions

1) Given a dataset of fixed size and fixed cluster:
   a) How does each library compare in terms of runtime & accuracy?
   b) What are the characteristics of the network usage of the libraries?

2) Given a dataset of fixed size:
   a) Does scale out reduce the runtime linearly as expected?
   b) Does the model accuracy vary with scale out?

3) Given a cluster with fixed number of worker nodes:
   How does each framework behave when the dataset size is increased linearly?

Case Study: Deep Dive into PCA Observations
RQ3: 3-Node Setup - Effect of Dataset Size (Runtime)
RQ3: 3-Node Setup - Effect of Dataset Size (Memory)

<table>
<thead>
<tr>
<th>Algorithm/DS Size</th>
<th>1G-SpML</th>
<th>1G-SW</th>
<th>2G-SpML</th>
<th>2G-SW</th>
<th>3G-SpML</th>
<th>3G-SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>37.3G</td>
<td>15.5G</td>
<td>41.9G</td>
<td>23.8G</td>
<td>48.5G</td>
<td>29.6G</td>
</tr>
<tr>
<td>K-Means Clustering</td>
<td>32.6G</td>
<td>14.7G</td>
<td>41.8G</td>
<td>22.4G</td>
<td>44.9G</td>
<td>29.6G</td>
</tr>
<tr>
<td>PCA</td>
<td>38.2G</td>
<td>22.1G</td>
<td>42.5G</td>
<td>26.9G</td>
<td>48.7G</td>
<td>30.8G</td>
</tr>
<tr>
<td>MLP</td>
<td>38.2G</td>
<td>22.3G</td>
<td>43.4G</td>
<td>27.4G</td>
<td>49.5G</td>
<td>31.8G</td>
</tr>
</tbody>
</table>
Research Questions

1) Given a dataset of fixed size and fixed cluster:
   a) How does each library compare in terms of runtime & accuracy?
   b) What are the characteristics of the network usage of the libraries?

2) Given a dataset of fixed size:
   a) Does scale out reduce the runtime linearly as expected?
   b) Does the model accuracy vary with scale out?

3) Given a cluster with fixed number of worker nodes:
   How does each framework behave when the dataset size is increased linearly?

4) Case Study: Deep Dive into PCA Observations
RQ4: 3 Node Setup - Vary “K” in PCA

Numerical Processing Library | Breeze | Matrix Toolkits Java
Answers: Research Questions

RQ1: Initial Looks
- Sparkling Water Accuracy ≥ Spark ML Accuracy (Except PCA)
- Sparkling Water Training Time < Spark ML Training Time (Except PCA)
- Sparkling Water Network Usage << Spark ML Network Usage

RQ2: Scale Out Effect
- Gain in runtime performance but algorithm dependent.
- Scale Out Does Not Affect Training/Testing Accuracy
Answers: Research Questions

RQ3: Dataset Size Increase Effect:
- Setup phase takes a hit due to data distribution & in-memory store/swap
- Training time increases linearly as dataset size increases
- Memory footprint for algos independent of space complexity (in-memory data)

RQ4: PCA - Varying “K” Effect:
- Sparkling Water takes substantially longer for training
- Sparkling Water yields larger (but diminishing) returns on accuracy
- Difference due to usage of different libraries
## Conclusions:

### Setup Runtime:
- **Less** due to lazy load of Spark Dataframe
- **More** due to *in-memory* load of Compressed H2O Frame

### Training Runtime:
- **More** for algos except PCA (large K). **More** affected by dataset size increase.
- **Less** for algos except PCA (large K). **Less** affected by dataset size increase.

### Memory Footprint:
- **More** due to decompression into Java objects in memory
- **Less** due to JIT decompression in CPU registers from memory

### Network Usage:
- **More** due to lazy load paradigm of Spark Dataframe
- **Less** due to the exploitation of data locality by algorithms

### Model Performance:
- Roughly *Equal* for LR, **Less** for MLP & PCA
- Roughly *Equal* for LR, **More** for MLP & PCA
Future Work

- **Feb**
  - Setup Data, Cluster & Decide RQ

- **March-Mid**
  - Frame & Conduct Experiments

- **Mid-End March**
  - Deep Dive Into Observations

- **1 Term**
  - Scale Out With Bigger Data, More Algos

- **2 Terms**
  - Test More Frameworks
Thank You!

Q & A

Thinking Out Loud ....

“Sparkling Water has more knobs that can be tuned for algorithms”

“Sparkling Water’s default hyperparameters per algorithm yields better performance over Spark ML”

“Log messages in Sparkling Water are more descriptive than Spark ML”

“User friendly guides from Sparkling Water”

“Ganglia lacks command line support for metrics collection”

“H2O.ai came out with H2O Deep Water (Distributed GPU Based DL) when we were doing our project!”