SAS vs. Netezza vs. Spark: Regression Benchmarks

Aaron Rodgers
2015-08-06
Outline

- Project Background
- Systems
  - SAS
  - Netezza
  - Spark
- Math
  - Linear Regression
  - Logistic Regression
- Benchmarks
  - Results
  - Interpretation


Background

- Spoke with a supervisor late June 2015 regarding Hadoop/Spark applications.
  - Using Hadoop for data ETL.
  - Using Spark for regression analysis.

- Supervisor's team has assembled a trial dataset for Coronary Artery Disease (CAD) regression analysis.
  - 3700 input variables
  - Considering using Spark (~7GB)
Over 3700 potential predictors were considered in the model development.
Which platform available to the company offers the best performance for performing regression on large datasets?
System: SAS

IBM P770 * 2

- Specs for 1 block (4 max):
  - 16 x 3.8 GHz POWER7+ cores.
  - Up to 1 TB of 1066 MHz DDR3.

- Price: ~$400K each.
System: Netezza (Dev)

Twinfin 3650

- Specs for 1 s-blade (max 12):
  - 4 cores
  - 16 GB RAM
  - dual-core FPGA

- Price: ~$1.2M.
System: Spark

M3.xlarge instance * 4

- Specs for 1 node:
  - 4 vCPU
  - 16 GB RAM
- Price: $200 / 6 days (all)
- Data files live in HDFS.
HDFS

Node 1

Node 2

Node 3

Node 4
HDFS + Spark

Node 1

Node 2

Node 3

Node 4
Task: Linear Regression

1. Plot points:

2. Fit a line:

\[ y = a + bx \]

Output var          Input var
intercept         coefficient

Regression finds the intercept and coefficients.

Cricket Chirps by Temp*

\[ y = \text{chirps / sec} \]

\[ x = \text{Temperature °C} \]

*not real data
Task: Multiple Linear Regression

1. Plot points:

2. Fit a plane:

$$y = a + b_1 x_1 + b_2 x_2$$
Task: Logistic Regression

Output variable is a probability, not a measurement.

Fitting a line gives bad probability predictions, so instead we fit a curve that stays between 0 and 1.

*not real data
Measuring Error

Measure the distance from each point to the solution line.

Square these distances and sum them. This is the **squared error**.
Suppose there are two parameters to fit: intercept $a$, coefficient $b$.

1. Choose some initial values for these parameters. Also, choose a step size.

2. Randomly select an observation from your sample and use it to estimate the gradient of the error surface at your current position. Move down the gradient one step size.

3. Repeat with the remaining observations.

4. If necessary, iterate over the whole dataset again.

By this process, we "descend" the error gradient. "Stochastic" refers to randomly selecting observations.
Generating Dummy Data

Sample line in a dummy data file:

<table>
<thead>
<tr>
<th>input_vars (random)</th>
<th>output_var</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>9433</td>
</tr>
</tbody>
</table>
Generating Dummy Data

Sample line in a dummy data file:

```
10  83  22  11          9433
```

$$100 \times 10 + 80 \times 83 + 60 \times 22 + 40 \times 11 + 200 + 33 = 9433$$

- `input_vars` (random)
- `output_var`
- `coefficients` (choose these)
- `intercept`
- `noise` (random)
Multiple Linear Regression in SAS

k10x4.dat:

<table>
<thead>
<tr>
<th>x_0000</th>
<th>x_0001</th>
<th>x_0003</th>
<th>x_0004</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>59</td>
<td>32</td>
<td>51</td>
<td>10246</td>
</tr>
<tr>
<td>17</td>
<td>66</td>
<td>89</td>
<td>26</td>
<td>13547</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>86</td>
<td>79</td>
<td>12528</td>
</tr>
<tr>
<td>71</td>
<td>38</td>
<td>75</td>
<td>12</td>
<td>15315</td>
</tr>
</tbody>
</table>

```sas
proc import
datafile="/path/to/k10x4.dat"
dbms=dlm
out=work.dat
replace;
delimiter='09'x;
/*delimiter=' ';*/
getnames=yes;
run;
```

```sas
proc reg data=work.dat;
   model y=x_;
run;
```

The REG Procedure
Model: MODEL1
Dependent Variable: y

<table>
<thead>
<tr>
<th>Number of Observations Read</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations Used</td>
<td>10000</td>
</tr>
</tbody>
</table>

Analysis of Variance

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>Mean Square</th>
<th>F Value</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>4</td>
<td>1.454177E11</td>
<td>36354428506</td>
<td>4.372E7</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Error</td>
<td>9995</td>
<td>8310230</td>
<td>831.43872</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>9999</td>
<td>1.45426E11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Root MSE: 28.83468
Dependent Mean: 15429
Adj R-Sq: 0.9999
Coeff Var: 0.18688

Parameter Estimates

| Variable | DF | Parameter Estimate | Standard Error | t Value | Pr > |t| |
|----------|----|-------------------|----------------|---------|------|-----|
| Intercept| 1  | 198.39580         | 1.24446        | 159.42  | <.0001 |
| x_0000   | 1  | 100.00068         | 0.01100        | 9087.69 | <.0001 |
| x_0001   | 1  | 80.00586          | 0.01116        | 7168.85 | <.0001 |
| x_0002   | 1  | 60.01498          | 0.01105        | 5428.95 | <.0001 |
| x_0003   | 1  | 39.99552          | 0.01114        | 3590.57 | <.0001 |
Benchmarks

- Use multiple linear regression
  - Linear regression and logistic regression are both solved with SGD, so doing linear instead of logistic still gives a meaningful benchmark.

- Data sizes:
  - k100x1200.dat 360MB
  - m1x1200.dat 3.4 GB
  - m3x1200.dat 11GB
  - m6x1200.dat 21GB

- Pick parameters for Spark
  - Step size
  - Num iterations
Results

But see Spark caveats!

(Memory Error)
Interpretation / Caveats

SAS handles like a champ, even with data sets ~20GB.

Good news for Spark:

- Spark machine allocation is elastic.
- At first glance, it seems Spark keeps up with SAS, but...

Bad news for Spark:

- Mean Squared Error is terrible! (worse by factor of ~10^9)
  - Spark must sacrifice significant accuracy in order to maintain time performance comparable to SAS.
  - Of course, this is somewhat expected given the disparity between the SAS hardware and Spark hardware.
  - More experimentation is needed...
- Spark doesn't pick step size for you, so you must experiment (incurring multiple runs).
- Must transfer data to AWS. (9.1MB/sec, or 1GB/110sec)
Other Unresolved Questions

- Did I pick the best cluster composition?
  - Many small nodes vs. a few large nodes?
  - Many large nodes?
- Security?
- Spark about to be outdated? X-Point Mem