Enabling R for Big Data with PL/R and PivotalR
Real World Examples on Hadoop & MPP Databases

Woo J. Jung
Principal Data Scientist
Pivotal Labs
All In On Open Source

Still can’t believe we did this. Truly exciting.
Data Science Toolkit

**KEY TOOLS**
- git
- MADlib
- IP[y]:
- RStudio

**KEY LANGUAGES**
- R
- python
- SQL
- Scala

**PLATFORM**
- Pivotal Cloud Foundry
- Pivotal Greenplum Database
- Pivotal HD
- Pivotal Big Data Suite
- Pivotal HAWQ
- Spark
- Spring XD
How Pivotal Data Scientists Select Which Tool to Use

Prototype in R or directly in MADlib/PivotalR

Is the algorithm of choice available in MADlib/PivotalR?

Yes
Build final set of models in MADlib/PivotalR

No
Do opportunities for explicit parallelization exist?

Yes
Build final set of models in PL/R

No
Connect to Pivotal via ODBC

Optimized for both algorithm efficiency and code overhead
Prototype in R or directly in MADlib/PivotalR

Is the algorithm of choice available in MADlib/PivotalR?

Yes
Build final set of models in MADlib/PivotalR

No
Do opportunities for explicit parallelization exist?

Yes
Build final set of models in PL/R

No
Connect to Pivotal via ODBC

Optimized for both algorithm efficiency and code overhead
MADlib: Toolkit for Advanced Big Data Analytics

- **Better Parallelism**
  - Algorithms designed to leverage MPP or Hadoop architecture
- **Better Scalability**
  - Algorithms scale as your data set scales
  - No data movement
- **Better Predictive Accuracy**
  - Using all data, not a sample, may improve accuracy
- **Open Source**
  - Available for customization and optimization by user

madlib.net
Elastic Net Regularization

This module implements elastic net regularization for linear and logistic regression problems.

Training Function

The training function has the following syntax:

```r
elastic_net_train( tbl_source, tbl_result, ml_rset, ml_rset_key, ml_rset_key_type, target_family, alpha, l2penalty, l1penalty, elastic_penalty, glm_penalty, optimizer, optimizer_params, excluded, exclude_null, tol)```

Arguments

- `tbl_source`:
  - TEXT: The name of the table containing the training data.

- `tbl_result`:
  - TEXT: Name of the generated table containing the output model. The output table produced by the `elastic_net_train()` function has the following columns:
    - `regress_family`: The regression type - 'gaussian' or 'binomial'.
    - `features`: An array of the features (independent variables) passed into the analysis.
    - `l1_penalty`: The penalty term for the selected features.
    - `l2_penalty`: The penalty term for all selected and unselected features.
    - `intercept`: Whether or not the intercept is included in the model.
    - `n_rounds`: The negative value of the first equation above (i.e. a constant depending on the data set).
    - `score_train`: Whether the score was normalized (standard deviation was 1)
    - `score_valid`: Whether the score was corrected for the training set.
    - `score_test`: Whether the score was corrected for the test set.
    - `tol`: The number of iterations executed.

- `col_dep_var`:
  - TEXT: An expression for the dependent variable.

  Both `col_dep_var` and `col_id_vars` must be valid PostgreSQL expressions. For example, `col_dep_var = 'log(y)'`, and `col_id_vars = 'array(x1, x2, 1)'`. In the binomial case, you can use a Boolean expression, for example, `col_dep_var = 'y = 0'`.

- `col_id_vars`:
  - TEXT: An expression for the independent variables. Use `*` to specify all columns of `tbl_source` except those listed in the excluded string. If `col_dep_var` is a column name, it is automatically excluded from the independent variables. However, if `col_dep_var` is a valid PostgreSQL expression, any column names used within the expression are only excluded if they are explicitly included in the excluded string. It is a good idea to add all column names involved in the dependent variable expression to the excluded string.

- `regress_family`:
  - TEXT: The regression type, either 'gaussian' (linear) or 'binomial' (logistic)
PivotalR: Bringing MADlib and HAWQ to a Familiar R Interface

• Challenge
  Want to harness the familiarity of R’s interface and the performance & scalability benefits of in-DB/in-Hadoop analytics

• Simple solution:
  Translate R code into SQL

```
SELECT madlib.linregr_train('houses', 'houses_linregr', 'price', 'ARRAY[1, tax, bath, size]');
```

**PivotalR**

```r

d <- db.data.frame("houses")
houses_linregr <- madlib.lm(price ~ tax + bath + size, data=d)
```

http://cran.r-project.org/web/packages/PivotalR/index.html
https://pivotalsoftware.github.io/gp-r/
https://github.com/pivotalsoftware/PivotalR
PivotalR Design Overview

- Call MADlib’s in-DB machine learning functions directly from R
- Syntax is analogous to native R function

1. R → SQL
2. SQL to execute
3. Computation results

- Data doesn’t need to leave the database
- All heavy lifting, including model estimation & computation, are done in the database
More Piggybacking

```r
plot_dt.madlib
function (x, uniform = FALSE, branch = 1, compress = FALSE, nspace,
  margin = 0, minbranch = 0.3, ...)
{
  library(rpart)
  class(x) <- "rpart"
  plot(x, uniform = uniform, branch = branch, compress = compress,
  nspace = nspace, margin = margin, minbranch = minbranch,
  ...)
}
<environment: namespace:PivotalR>

fit <- madlib.rpart(rings < 10 ~ length + diameter + height + whole + shell,
  data=x, parms = list(split='gini'), control = list(cp=0.005))
plot(fit)
text(fit)
```
How Pivotal Data Scientists Select Which Tool to Use

Prototype in R or directly in MADlib/PivotalR

Is the algorithm of choice available in MADlib/PivotalR?

Yes

Build final set of models in MADlib/PivotalR

No

Do opportunities for explicit parallelization exist?

Yes

Build final set of models in PL/R

No

Connect to Pivotal via ODBC

Optimized for both algorithm efficiency and code overhead
What is Data Parallelism?

• Little or no effort is required to break up the problem into a number of parallel tasks, and there exists no dependency (or communication) between those parallel tasks.

• Also known as ‘explicit parallelism’

• Examples:
  – Have each person in this room weigh themselves: Measure each person’s weight in parallel
  – Count a deck of cards by dividing it up between people in this room: Count in parallel
  – MapReduce
  – apply() family of functions in R
Procedural Language R (PL/R)

• Parallelized model building in the R language
• Originally developed by Joe Conway for PostgreSQL
• Parallelized by virtue of piggybacking on distributed architectures

http://pivotalsoftware.github.io/gp-r/
Parallelized Analytics in Pivotal via PL/R: An Example

• Parsimonious – R piggy-backs on Pivotal’s parallel architecture
• Minimize data movement
• Build predictive model for each state in parallel
Parallelized R via PL/R: One Example of Its Use

- With placeholders in SQL, write functions in the native R language
- Accessible, powerful modeling framework

```sql
-- Create TYPE to store model results
CREATE TYPE lm_type AS (Variable text, Coef_Est float, Std_Error float, T_Stat float, P_Value float);

-- Create PL/R function
CREATE FUNCTION lm(wage float8[], rentshouse float8[], married float8[]) RETURNS SETOF lm_type AS
$$
ml<- lm(wage-rentshouse + married)
m1_s<- summary(ml)$coef
temp_ml<- data.frame(rownames(m1_s), m1_s)
return(temp_ml)
$$
LANGUAGE 'plr';
```
Parallelized R via PL/R: One Example of Its Use

- Execute PL/R function

```sql
--Run PL/R function
SELECT h_state, `lm(wage, rentshouse, married)`.* FROM use_r.census1_array_state;
```

Plain and simple table is returned
Examples of Usage
Pivotal Data Science: Areas of Expertise

Energy
Retail
FSI
Life Science / Healthcare
Manufacturing
Communications

TEXT ANALYTICS
SECURITY & FRAUD
DIGITAL MEDIA
IMAGE / VIDEO
GRAPH / NETWORK
Pivotal Data Science: Packaged Services

LAB PRIMER
(2-Week Roadmapping)
• Analytics Roadmap
• Prioritized Opportunities
• Architectural Recommendations

DATA JAM
(Internal DS Contest)
• Hands-on training
• Hosted data on Pivotal Data stack
• Results review & assessment

LAB 100
(Analytics Bundle)
• On-site MPP analytics training
• Analytics toolkit
• Quick insight (2 weeks)

LAB 600
(6-Week Lab)
• Prof. services
• Data science model building
• Ready-to-deploy model(s)

LAB 1200
(12-Week Lab)
• Prof. services
• Data science model building
• Ready-to-deploy model(s)
The Internet of Things: Smart Meter Analytics
Engagement Summary

• Objective
  – Build key foundations of a data-driven framework for anomaly detection to leverage in revenue protection initiatives

• Results
  – With limited access to limited data, our models (FFT and Time Series Analysis) identified 191K potentially anomalous meters (7% of all meters).

• High Performance
  – Pivotal Big Data Suite including MADlib and PL/R
  – 90 seconds to compute FFT for over 3.1 million meters (~3.5 billion readings) \( \Rightarrow 0.0288 \text{ ms/meter} \)
  – ~36 minutes to compute time series models for over 3.1 million meters (~3.5 billion readings) \( \Rightarrow 0.697 \text{ ms/meter} \)
Anomaly Detection Methodology & Results

All Data (4.5 million meters / ~20 billion meter readings)

Step 1: Select Data for Advanced Modeling (3.1 million meters / ~3.5 billion meter readings)

Step 2: Detect Anomalies From Frequency Domain Analysis (547K)

Step 3: Detect Anomalies From Time Domain Analysis (485K)

Step 4: Detect Anomalies From Combined Analysis (191K)
-- create type to store frequency, spec, and max freq
create type fourier_type AS (freq text, spec text, freq_with_maxspec float8);

-- create plr function to compute periodogram and return frequency with maximum spectral density
create or replace function pgram_concise(tsval float8[]) RETURNS float8 AS $$
  rpgram <- spec.pgram(tsval, fast=FALSE, plot=FALSE, detrend=TRUE)
  freq_with_maxspec <- rpgram$freq[which(rpgram$spec==max(rpgram$spec))]
  return(freq_with_maxspec)
$$ LANGUAGE 'plr';

-- execute function
create table pg_gram_results
as select geo_id, meter_id, pgram_concise(load_ts) FROM meter_data distributed by (geo_id,meter_id);
Most Households Use Energy in Daily or Half-Daily Cycles

Estimated Periodicity of Meters

- Dominant periodicity (i.e. maximum frequency) of each meter is computed
- ~80% of all households show daily or half-daily patterns of energy usage
- ~20% of all households show anomalous patterns of energy usage
- Flag meters falling into the 20% as potentially anomalous meters
- Follow-up Items: Event type of the anomalies w.r.t. Revenue Protection to be determined with additional data & models
Irregular Patterns of Energy Consumption Displayed by Detected Anomalous Meters

FFT Analysis: Time Series of an Ordinary Meter

FFT Analysis: Time Series of an Anomalous Meter
Parallelize the Generation of Visualizations
Parallelize Visualization Generation

-- create function
create or replace function plot_pva_plr(brand text, gender text, department int4, location text, week_agg date[], actunal_agg float8[], predicted_agg float8[], wmape float8, r2 float8)
returns float8 as
$$
t2<- as.Date(week_agg)
pdf(paste("/home/gpadmin/wjung/plots_pva/", brand, ",", gender, ",", department, ",", location,
",.pdf", sep=""), width=21, height=10)

# set plotting window size
par(mar=c(4, 5, 4, 5), mtext=c(1,1))

# plot 1st series - actual units
plot(actual_agg~t2, xaxt="n", type="o", main=paste("brand_dept=",department[1], ",", location="", location[1], ",", weighted mape="", round(wmape[1,2]), xlab="", ylab="", cex=.7, col="red", axes=F)
axis(2, ylim=c(0,max(actual_agg)), lwd=2)
mtext(2, text="Actual Units", line=2, col="red")

# plot 2nd series - predicted units
par(new=T)
plot(predicted_agg~t2, xaxt="n", type="o", main=paste("brand_dept=",department[1], ",", location="", location[1], ",", weighted mape="", round(wmape[1,2]), xlab="", ylab="", cex=.7, col="blue", axes=F)
axis(4, ylim=c(0,max(actual_agg)), lwd=2)
mtext(4, text="Predicted Units", line=2, col="blue")

# plot x axis
axis(1, t2, format(t2, "%y %b %d ") , cex.axis=,6, lwd=2)

dev.off()
$$
language 'plr';

--run function
select plot_pva_plr(brand, gender, department, location,
week_agg, actual_agg, predicted_agg, wmape, r2) from pva_filtered_array;
Parallelize Visualization Generation
Demand Modeling & What-If Scenario Analysis
Scalable Algorithm Development Using R
Prototyping Dashboards on RShiny
Engagement Overview

Customer’s Business Goal

- Make **data-driven decisions** about how to allocate resources for planning & inventory management

- Compose rich set of **reusable data assets from disparate LOBs** and make available for ongoing analysis & reporting

- Build **parallelized demand models for 100+ products & locations**

- Develop **scalable** Hierarchical/Multilevel Bayesian Modeling algorithm (Gibbs Sampling)

- Construct framework & prototype app for **what-if scenario analysis** in RShiny
Overview of Hierarchical Linear Model

Likelihood

\[ y_i \sim N(X_iB_j[i], \sigma^2_y), \text{ for } i = 1, \ldots, n \]

\[ B_j \sim N(\mu_\beta, V_\beta), \text{ for } j = 1, \ldots, J, \]

\[ h \sim G(s^{-2}, \nu) \]

Priors for parameters

Priors for hyperparameters

\[ \mu_\beta \sim N(\mu_{\beta}, \Sigma_{\beta}) \]

\[ V_\beta^{-1} \sim W(\nu_{\beta}, V^{-1}_{\beta}) \]

Posterior

\[ \propto \text{Likelihood} \times \text{Priors for parameters} \times \text{Priors for hyperparameters} \]

This joint posterior distribution does not take the form of a known probability density, thus it is a challenge to draw samples from it directly.

However, the full conditional posterior distributions follow known probability densities (Gibbs Sampling).
Game Plan

1. Figure out which components of the Gibbs Sampler can be “embarrassingly” parallelized, i.e. the key building blocks
   - Mostly matrix algebra calculations & draws from full conditional distributions, parallelized by Product-Location

2. Build functions (i.e. in PL/R) for each of the building blocks

3. Build a “meta-function” that ties together each of the functions in (2) to run a Gibbs Sampler

4. Run functions for K iterations, monitor convergence, summarize results
Examples of Building Block Functions

-- Function to draw Vbeta_inv from Wishart dist'n
create or replace function Vbeta_inv_draw(float8, float8[])
returns float8[] as
$$
\text{library(MCMCpack)}
\text{return(rwish(arg1,arg2))}
$$
language 'plr';

-- Function to compute mean pooled coefficient vector to use in drawing a new pooled coefficient vector. This function allows for user-specified priors on the coefficients. For use at highest level of the hierarchy.
create or replace function beta_mu_prior(float8[], float8[], float8[], float8[], float8[])
returns float8[] as
$$
beta_mu<- \text{arg1} \times \text{arg2} \times \text{arg3} + \text{as.matrix(arg4} \times \text{arg5})
\text{return(beta_mu)}
$$
language 'plr';

-- Function to draw new beta. Takes mean vector of the multivariate normal distribution as its first parameter, and the variance matrix of the multivariate normal distribution as its second parameter. Used in cases where beta_1 and beta_mu are drawn.
create or replace function beta_draw(float8[], float8[])
returns float8[] as
$$
\text{library(MASS)}
beta_mu_draw<- \text{rmvnorm}(1, \text{arg1}, \text{arg2})
\text{return(beta_mu_draw)}
$$
language 'plr';
Meta-Function & Execution

-- Get the first draw from the Gibbs Sampler. Note that this drops all existing tables storing previous runs of the Gibbs Sampler for a given model_name. Supply a new model_name to preserve older models.
select * from gibbs_init(
  'ml'
  , 'd'
  , 'location'
  , 'department'
  , 14
  , 15
  , 'sales'
  , 'array[1, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14]'
  , 'array[0,0,0,0,0,0,0,0,0,0,0,0,0,0]'
  , 'array[1,1,1,1,1,1,1,1,1,1,1,1,1,1]'
  , 1
  , 'random'
);

-- Update Gibbs samples. Select beginning & ending number of iterations.
select * from gibbs(
  'ml'
  , 'd'
  , 'location'
  , 'department'
  , 14
  , 15
  , 'sales'
  , 'array[1, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14]'
  , 'array[0,0,0,0,0,0,0,0,0,0,0,0,0,0]'
  , 'array[1,1,1,1,1,1,1,1,1,1,1,1,1,1]'
  , 1
  , 2
  , 100000);
PivotalR & RShiny

- Data doesn’t need to leave the database
- All heavy lifting, including model estimation & computation, are done in the database

SQL to execute

Computation results

RPostgreSQL

RShiny

Data lives here

Database/Hadoop w/ MADlib
RShiny Server

No data here

14.49% increase in simulated sales from last year

New simulated sales: 8771
Last year simulated sales: 7661
Last year actual sales: 7780
Simulated sales + model err: 8590

What-If Scenario Analysis

Select parameters for the what-if scenario.

Brand: Brand 1
Gender: Female
Department: WOMENS TEES
Store Type: Outlet
Scenario Prediction Period: 09/01/14 - 12/31/14
View decomp plot per: Quarter

Fast Update Driver:
% Items on Promo

Increase/Decrease Amount:
0.5

<table>
<thead>
<tr>
<th>Week</th>
<th>Assortment</th>
<th>Price</th>
<th>% Items on Promo</th>
<th>% Items on Redline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-08-31</td>
<td>7</td>
<td>9.37</td>
<td>0.5</td>
<td>0.66</td>
</tr>
<tr>
<td>2014-08-07</td>
<td>7</td>
<td>9.13</td>
<td>0.5</td>
<td>0.64</td>
</tr>
<tr>
<td>2014-08-14</td>
<td>8</td>
<td>9.49</td>
<td>0.5</td>
<td>0.63</td>
</tr>
<tr>
<td>2014-09-21</td>
<td>8</td>
<td>9.53</td>
<td>0.5</td>
<td>0.62</td>
</tr>
<tr>
<td>2014-09-28</td>
<td>10</td>
<td>9.92</td>
<td>0.5</td>
<td>0.62</td>
</tr>
<tr>
<td>2014-10-05</td>
<td>10</td>
<td>10.1</td>
<td>0.5</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Next Steps

• Continue to build even more PivotalR wrapper functions
• Identify more areas where core R functions can be re-leveraged and made scalable via PivotalR
• Explore, learn, and share notes with other packages like PivotalR
• Explore closer integration with Spark, MLlib, H20
• PL/R wrappers directly from R
Thank You

Have Any Questions?
Check out the Pivotal Data Science Blog!
http://blog.pivotal.io/data-science-pivotal
Additional References

• PivotalR
  - http://cran.r-project.org/web/packages/PivotalR/PivotalR.pdf
  - https://github.com/pivotalsoftware/PivotalR
  - Video Demo

• PL/R & General Pivotal+R Interoperability
  - http://pivotalsoftware.github.io/gp-r/

• MADlib
  - http://madlib.net/
  - http://doc.madlib.net/latest/