Best Practices for Building and Deploying Predictive Models Over Big Data

Robert Grossman
Open Data Group & Univ. of Chicago

Collin Bennett
Open Data Group

October 23, 2012
1. Introduction
2. Building Predictive Models – EDA, Features
3. Case Study: MalStone
4. Deploying Predictive Models
5. Working with Multiple Models
6. Case Study: CTA
7. Building Models over Hadoop Using R
8. Case Study: Building Trees over Big Data
9. Analytic Operations – SAMS
10. Case Study: AdReady
11. Building Predictive Models – Lift of a Model
12. Case Study: Matsu

The tutorial is divided into 12 Modules. You can download all the modules and related materials from tutorials.opendatagroup.com.
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 1: Introduction

Robert Grossman
Open Data Group & Univ. of Chicago

Collin Bennett
Open Data Group

October 23, 2012
What Is Small Data?

• 100 million movie ratings
• 480 thousand customers
• 17,000 movies
• From 1998 to 2005
• Less than 2 GB data.
• Fits into memory, but very sophisticated models required to win.
What is Big Data?

- The leaders in big data measure data in Megawatts.
  - As in, Facebook’s leased data centers are typically between 2.5 MW and 6.0 MW.
  - Facebook’s new Pineville data center is 30 MW.

- At the other end of the spectrum, databases can manage the data required for many projects.
An algorithm and computing infrastructure is “big-data scalable” if adding a rack of data (and corresponding processors) allows you to compute over more data in the same time.
What is Predictive Analytics?

**Short Definition**

- Using data to make decisions.

**Longer Definition**

- Using data to take actions and make decisions using models that are statistically valid and empirically derived.
Key Modeling Activities

- Exploring data analysis – goal is to understand the data so that features can be generated and model evaluated
- Generating features – goal is to shape events into meaningful features that enable meaningful prediction of the dependent variable
- Building models – goal is to derive statistically valid models from a learning set
- Evaluating models – goal is to develop meaningful report so that the performance of one model can be compared to another model
Analytic algorithms & models

Analytic operations, security & compliance

Analytic strategy & governance
IT Organization

Analytic Infrastructure
What Is Analytic Infrastructure?

Analytic infrastructure refers to the software components, software services, applications and platforms for managing data, processing data, producing analytic models, and using analytic models to generate alerts, take actions and make decisions.
Build models

Analytic models & algorithms

Modeling Group
Building models:

Data → Model Producer → Model

CART, SVM, k-means, etc. → PMML

R icon
Operations, product

Analytic Operations
Deployed models
Life Cycle of Predictive Model

- **Exploratory Data Analysis (R)**
- **Build model in dev/modeling environment**
- **Process the data (Hadoop)**
- **Refine model**
- **Deploy model in operational systems with scoring application (Hadoop, streams & Augustus)**
- **Retire model and deploy improved model**

PMML

Operations

Log files
## Key Abstractions

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Credit Card Fraud</th>
<th>On line advertising</th>
<th>Change Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>Credit card transactions</td>
<td>Impressions and clicks</td>
<td>Status updates</td>
</tr>
<tr>
<td>Entities</td>
<td>Credit card accounts</td>
<td>Cookies, user IDs</td>
<td>Computers, devices, etc.</td>
</tr>
<tr>
<td>Features (examples)</td>
<td># transactions past n minutes</td>
<td># impressions per category past n minutes, RFM related, etc.</td>
<td># events in a category previous time period</td>
</tr>
<tr>
<td>Models</td>
<td>CART</td>
<td>Clusters, trees, recommendation, etc.</td>
<td>Baseline models</td>
</tr>
<tr>
<td>Scores</td>
<td>Indicates likelihood of fraudulent account</td>
<td>Likelihood of clicking</td>
<td>Indicates likelihood that a change has taken place</td>
</tr>
</tbody>
</table>
Event Based Updating of Feature Vectors

Event loop:
1. take next event
2. update one or more associated FVs
3. rescore updated FVs to compute alerts
Example Architecture

• Data lives in Hadoop Distributed File System (HDFS) or other distributed file systems.
• Hadoop streams and MapReduce are used to process the data.
• R is used for Exploratory Data Analysis (EDA) and models are exported as PMML.
• PMML models are imported and Hadoop, Python Streams and Augustus are used to score data in production.
Questions?

For the most current version of these slides, please see tutorials.opendatagroup.com
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 2: Building Models – Data Exploration and Features Generation

Robert Grossman
Open Data Group & Univ. of Chicago

Collin Bennett
Open Data Group

October 23, 2012
Key Modeling Activities

- Exploring data analysis – goal is to understand the data so that features can be generated and model evaluated
- Generating features – goal is to shape events into meaningful features that enable meaningful prediction of the dependent variable
- Building models – goal is to derive statistically valid models from learning set
- Evaluating models – goal is to develop meaningful report so that the performance of one model can be compared to another model
Naïve Bayes

K nearest neighbor

1957 K-means

1977 EM Algorithm

1984 CART

1993 C4.5

1994 A Priori

1995 SVM

1997 AdaBoost

1998 PageRank

Beware of any vendor whose unique value proposition includes some “secret analytic sauce.”

Pessimistic View: We Get a Significantly New Algorithm Every Decade or So

1970’s    neural networks
1980’s    classifications and regression trees
1990’s    support vector machine
2000’s    graph algorithms
In general, understanding the data through exploratory analysis and generating good features is much more important than the type of predictive model you use.
Questions About Datasets

• Number of records
• Number of data fields
• Size
  – Does it fit into memory?
  – Does it fit on a single disk or disk array?
• How many missing values?
• How many duplicates?
• Are there labels (for the dependent variable)?
• Are there keys?
Questions About Data Fields

• Data fields
  – What is the mean, variance, ...?
  – What is the distribution? Plot the histograms.
  – What are extreme values, missing values, unusual values?

• Pairs of data fields
  – What is the correlation? Plot the scatter plots.

• Visualize the data
  – Histograms, scatter plots, etc.
```r
> a.data <- iris
> summary(a.data)

<table>
<thead>
<tr>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>4.300</td>
<td>Min.</td>
<td>1.000</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>5.100</td>
<td>1st Qu.</td>
<td>1.600</td>
</tr>
<tr>
<td>Median</td>
<td>5.800</td>
<td>Median</td>
<td>4.350</td>
</tr>
<tr>
<td>Mean</td>
<td>5.843</td>
<td>Mean</td>
<td>3.758</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>6.400</td>
<td>3rd Qu.</td>
<td>5.100</td>
</tr>
<tr>
<td>Max.</td>
<td>7.900</td>
<td>Max.</td>
<td>6.900</td>
</tr>
<tr>
<td></td>
<td>Max.</td>
<td></td>
<td>Max.</td>
</tr>
</tbody>
</table>
```
Computing Count Distributions

def valueDistrib(b, field, select=None):
    # given a dataframe b and an attribute field,
    # returns the class distribution of the field
    ccdict = {}
    if select:
        for i in select:
            count = ccdict.get(b[i][field], 0)
            ccdict[b[i][field]] = count + 1
    else:
        for i in range(len(b)):
            count = ccdict.get(b[i][field], 0)
            ccdict[b[i][field]] = count + 1
    return ccdict
Features

• Normalize
  – [0, 1] or [-1, 1]
• Aggregate
• Discretize or bin
• Transform continuous to continuous or discrete to discrete
• Compute ratios
• Use percentiles
• Use indicator variables
Predicting Office Building Renewal Using Text-based Features

<table>
<thead>
<tr>
<th>Classification</th>
<th>Avg R</th>
</tr>
</thead>
<tbody>
<tr>
<td>imports</td>
<td>5.17</td>
</tr>
<tr>
<td>clothing</td>
<td>4.17</td>
</tr>
<tr>
<td>van</td>
<td>3.18</td>
</tr>
<tr>
<td>fashion</td>
<td>2.50</td>
</tr>
<tr>
<td>investment</td>
<td>2.42</td>
</tr>
<tr>
<td>marketing</td>
<td>2.38</td>
</tr>
<tr>
<td>oil</td>
<td>2.11</td>
</tr>
<tr>
<td>air</td>
<td>2.09</td>
</tr>
<tr>
<td>system</td>
<td>2.06</td>
</tr>
<tr>
<td>foundation</td>
<td>2.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classification</th>
<th>Avg R</th>
</tr>
</thead>
<tbody>
<tr>
<td>technology</td>
<td>2.03</td>
</tr>
<tr>
<td>apparel</td>
<td>2.03</td>
</tr>
<tr>
<td>law</td>
<td>2.00</td>
</tr>
<tr>
<td>casualty</td>
<td>1.91</td>
</tr>
<tr>
<td>bank</td>
<td>1.88</td>
</tr>
<tr>
<td>technologies</td>
<td>1.84</td>
</tr>
<tr>
<td>trading</td>
<td>1.82</td>
</tr>
<tr>
<td>associates</td>
<td>1.81</td>
</tr>
<tr>
<td>staffing</td>
<td>1.77</td>
</tr>
<tr>
<td>securities</td>
<td>1.77</td>
</tr>
</tbody>
</table>
Twitter Intention & Prediction

• Use machine learning classification techniques:
  – Support Vector Machines
  – Trees
  – Naïve Bayes

• Models require clean data and lots of hand-scored examples
What is the Size of Your Data?

• Small
  – Fits into memory

• Medium
  – Too large for memory
  – But fits into a database

• Large
  – To large for a database
  – But you can use NoSQL database (HBase, etc.)
  – Or DFS (Hadoop DFS)
If You Can Ask For Something

- Ask for more data
- Ask for “orthogonal data”
Questions?

For the most current version of these slides, please see tutorials.opendatagroup.com
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 3: Case Study - MalStone

Robert Grossman
Open Data Group & Univ. of Chicago
Collin Bennett
Open Data Group

October 23, 2012
Part 1. Statistics over Log Files

• Log files are everywhere
• Advertising systems
• Network and system logs for intrusion detection
• Health and status monitoring
What are the Common Elements?

• Time stamps
• Sites
  – e.g. Web sites, computers, network devices
• Entities
  – e.g. visitors, users, flows
• Log files fill disks, many, many disks
• Behavior occurs at all scales
• Want to identify phenomena at all scales
• Need to group “similar behavior”
• Need to do statistics (not just sorting)
Abstract the Problem Using Site-Entity Logs

<table>
<thead>
<tr>
<th>Example</th>
<th>Sites</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measuring online advertising</td>
<td>Web sites</td>
<td>Consumers</td>
</tr>
<tr>
<td>Drive-by exploits</td>
<td>Web sites</td>
<td>Computers (identified by cookies or IP)</td>
</tr>
<tr>
<td>Compromised systems</td>
<td>Compromised computers</td>
<td>User accounts</td>
</tr>
</tbody>
</table>
MalStone Schema

- Event ID
- Time stamp
- Site ID
- Entity ID
- Mark (categorical variable)
- Fit into 100 bytes
Toy Example

Events collected by device or processor in time order

Map events by site

For each site, compute counts and ratios of events by type
Distributions

• Tens of millions of sites
• Hundreds of millions of entities
• Billions of events
• Most sites have a few number of events
• Some sites have many events
• Most entities visit a few sites
• Some visitors visit many sites
MalStone B

sites

entities

d_{k-2} \quad d_{k-1} \quad d_k

time
The Mark Model

• Some sites are marked (percent of mark is a parameter and type of sites marked is a draw from a distribution)

• Some entities become marked after visiting a marked site (this is a draw from a distribution)

• There is a delay between the visit and the when the entity becomes marked (this is a draw from a distribution)

• There is a background process that marks some entities independent of visit (this adds noise to problem)
Exposure Window

Monitor Window

d_{k-2} 

d_{k-1} 

d_{k} 

time
Notation

- Fix a site $s[j]$
- Let $A[j]$ be entities that transact during ExpWin and if entity is marked, then visit occurs before mark
- Let $B[j]$ be all entities in $A[j]$ that become marked sometime during the MonWin
- Subsequent proportion of marks is
B[j, t] are entities that become marked during MonWin[j]

d_{k-2}  d_{k-1}  d_{k}  

MonWin 1
MonWin 2
ExpWin

Time 50
Part 2. MalStone Benchmarks

MalGen and MalStone implementations are open source
MalStone Benchmark

- Benchmark developed by Open Cloud Consortium for clouds supporting data intensive computing.
- Code to generate synthetic data required is available from code.google.com/p/malgen
- Stylized analytic computation that is easy to implement in MapReduce and its generalizations.
# MalStone A & B

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Statistic</th>
<th># records</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MalStone A-10</td>
<td>r[j]</td>
<td>10 billion</td>
<td>1 TB</td>
</tr>
<tr>
<td>MalStone A-100</td>
<td>r[j]</td>
<td>100 billion</td>
<td>10 TB</td>
</tr>
<tr>
<td>MalStone A-1000</td>
<td>r[j]</td>
<td>1 trillion</td>
<td>100 TB</td>
</tr>
<tr>
<td>MalStone B-10</td>
<td>r[j, t]</td>
<td>10 billion</td>
<td>1 TB</td>
</tr>
<tr>
<td>MalStone B-100</td>
<td>r[j, t]</td>
<td>100 billion</td>
<td>10 TB</td>
</tr>
<tr>
<td>MalStone B-1000</td>
<td>r[j, t]</td>
<td>1 trillion</td>
<td>100 TB</td>
</tr>
</tbody>
</table>
• MalStone B running on 10 Billion 100 byte records
• Hadoop version 0.18.3
• 20 nodes in the Open Cloud Testbed
• MapReduce required 799 minutes
• Hadoop streams required 142 minutes
## Importance of Hadoop Streams

<table>
<thead>
<tr>
<th></th>
<th>MalStone B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop v0.18.3</td>
<td>799 min</td>
</tr>
<tr>
<td>Hadoop Streaming v0.18.3</td>
<td>142 min</td>
</tr>
<tr>
<td>Sector/Sphere</td>
<td>44 min</td>
</tr>
<tr>
<td># Nodes</td>
<td>20 nodes</td>
</tr>
<tr>
<td># Records</td>
<td>10 Billion</td>
</tr>
<tr>
<td>Size of Dataset</td>
<td>1 TB</td>
</tr>
</tbody>
</table>
Questions?

For the most current version of these slides, please see tutorials.opendatagroup.com
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 4: Deploying Analytics

Robert Grossman
Open Data Group & Univ. of Chicago

Collin Bennett
Open Data Group

October 23, 2012
Deploying analytic models

Analytic algorithms & models

Analytic Diamond

Analytic Infrastructure

Analytic operations, security & compliance
Problems with Current Techniques

• Analytics built in modeling group and deployed by IT.
• Time required to deploy models and to integrate models with other applications can be long.
• Models are deployed in proprietary formats
• Models are application dependent
• Models are system dependent
• Models are architecture dependent
Model Independence

• A model should be independent of the environment:
  o Production / Staging / Development
  o MapReduce / Elastic MapReduce / Standalone
  o Developer’s desktop / Analyst’s laptop
  o Production Workflow / Historical Data Store

• A model should be independent of coding language.
(Very Simplified) Architectural View

• The Predictive Model Markup Language (PMML) is an XML language for statistical and data mining models (www.dmg.org).

• With PMML, it is easy to move models between applications and platforms.
Predictive Model Markup Language (PMML)

• Based on XML
• Benefits of PMML
  – Open standard for Data Mining & Statistical Models
  – Provides independence from application, platform, and operating system
• PMML Producers
  – Some applications create predictive models
• PMML Consumers
  – Other applications read or consume models
PMML

• PMML is the leading standard for statistical and data mining models and supported by over 20 vendors and organizations.

• With PMML, it is easy to develop a model on one system using one application and deploy the model on another system using another application

• Applications exist in R, Python, Java, SAS, SPSS ...

Vendor Support

- SPSS
- SAS
- IBM
- Microstrategy
- Salford
- Tibco
- Zementis
- & other vendors

- R
- Augustus
- KNIME
- & other open source applications
Philosophy

• Very important to understand what PMML is not concerned with ...

• PMML is a specification of a model, not an implementation of a model

• PMML allows a simple means of binding parameters to values for an agreed upon set of data mining models & transformations

• Also, PMML includes the metadata required to deploy models
PMML Document Components

- Data dictionary
- Mining schema
- Transformation Dictionary
- Multiple models, including segments and ensembles.
- Model verification, ...
- Univariate Statistics (ModelStats)
- Optional Extensions
PMML Models

- trees
- associations
- neural nets
- naïve Bayes
- sequences
- text models
- support vector machines
- ruleset
- polynomial regression
- logistic regression
- general regression
- center based clusters
- density based clusters
PMML Data Flow

• Data Dictionary defines data
• Mining Schema defines specific inputs (MiningFields) required for model
• Transformation Dictionary defines optional additional derived fields
• Attributes of feature vectors are of two types
  – attributes defined by the mining schema
  – derived attributes defined via transformations
• Models themselves can also support certain transformations
• PMML also supports XML elements to describe data preprocessing.
Three Important Interfaces

Modeling Environment

Data \[\rightarrow\] Data Pre-processing \[\rightarrow\] Model Producer \[\rightarrow\] PMML Model

Deployment Environment

Model Consumer \[\rightarrow\] PMML Model \[\rightarrow\] Post Processing \[\rightarrow\] actions

data \[\rightarrow\] Model Consumer \[\rightarrow\] scores
Single Model PMML

PMML Document (Global)

Model ABC

Model Execution

Processing of elements for PMML supported models, such as trees, regression, Naive Bayes, baselines, etc.

Order of Evaluation:
DD → MS → TD → LT → Model

< 71 - PMML Field Diagram (v8)
Augustus PMML Producer

various data formats, including csv files, http streams, & databases

Producer Schema (part of PMML)

Model Producer

PMML Model

Producer Configuration File

Augustus (augustus.googlecode.com)
Augustus PMML Consumer

PMML Model

Model Consumer

Post Processing

scores

Consumer Configuration File

actions, alerts, reports, etc.

Augustus (augustus.googlecode.com)

various data formats, including csv files, http streams, & databases
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 5: Multiple Models

Robert Grossman
Open Data Group & Univ. of Chicago

Collin Bennett
Open Data Group

October 23, 2012
1. Trees

classification Trees

<table>
<thead>
<tr>
<th>Petal Len.</th>
<th>Petal Width</th>
<th>Sepal Len.</th>
<th>Sepal Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>14</td>
<td>33</td>
<td>50</td>
<td>A</td>
</tr>
<tr>
<td>24</td>
<td>56</td>
<td>31</td>
<td>67</td>
<td>C</td>
</tr>
<tr>
<td>23</td>
<td>51</td>
<td>31</td>
<td>69</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>45</td>
<td>28</td>
<td>57</td>
<td>B</td>
</tr>
</tbody>
</table>

- Want a function $Y = g(X)$, which predicts the red variable $Y$ using one or more of the blue variables $X[1], \ldots, X[4]$
- Assume each row is classified A, B, or C
Trees partition the feature space into regions by asking whether an attribute is less than a threshold.
2. Multiple Models: Ensembles
Key Idea: Combine Weak Learners

- Average several weak models, rather than build one complex model.
## Combining Weak Learners

<table>
<thead>
<tr>
<th>1 Classifier</th>
<th>3 Classifiers</th>
<th>5 Classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>55%</td>
<td>57.40%</td>
<td>59.30%</td>
</tr>
<tr>
<td>60%</td>
<td>64.0%</td>
<td>68.20%</td>
</tr>
<tr>
<td>65%</td>
<td>71.00%</td>
<td>76.50%</td>
</tr>
</tbody>
</table>

[Diagram of numbers arranged in a pattern]
Ensembles

- Ensembles are collections of classifiers with a rule for combining the results.
- The simplest combining rule:
  - Use average for regression trees.
  - Use majority voting for classification.
- Ensembles are unreasonably effective at building models.
Building ensembles over clusters of commodity workstations has been used since the 1990’s.
Ensemble Models

1. Partition data and scatter
2. Build models (e.g. tree-based model)
3. Gather models
4. Form collection of models into ensemble (e.g. majority vote for classification & averaging for regression)
Example: Fraud

• Ensembles of trees have proved remarkably effective in detecting fraud
• Trees can be very large
• Very fast to score
• Lots of variants
  – Random forests
  – Random trees
• Sometimes need to reverse engineer reason codes.
3. CUSUM

Assume that we have a mean and standard deviation for both distributions.

\[ f_0(x) = \text{baseline density} \]
\[ f_1(x) = \text{observed density} \]
\[ g(x) = \log(f_1(x)/f_0(x)) \]
\[ Z_0 = 0 \]
\[ Z_{n+1} = \max\{Z_n + g(x), 0\} \]
Alert if \( Z_{n+1} > \text{threshold} \)
4. Cubes of Models
Example 1: Cubes of Models for Change Detection for Payments System

Build separate segmented model for every entity of interest

Divide & conquer data (segment) using multidimensional data cubes

For each distinct cube, establish separate baselines for each quantify of interest

Detect changes from baselines

Estimate separate baselines for each quantify of interest
Example 2 – Cubes of Models for Detecting Anomalies in Traffic

• Highway Traffic Data
  – each day (7) x each hour (24) x each sensor (hundreds) x each weather condition (5) x each special event (dozens)
  – 50,000 baselines models used in current testbed
5. Multiple Models in Hadoop

- Trees in Mapper
- Build lots of small trees (over historical data in Hadoop)
- Load all of them into the mapper
- Mapper
  - For each event, score against each tree
  - Map emits
    - Key \( \text{value} = \text{event id} \text{ \& tree id, score} \)
- Reducer
  - Aggregate scores for each event
  - Select a “winner” for each event
Questions?

For the most current version of these slides, please see tutorials.opendatagroup.com
Best Practices for Building and Deploying Predictive Models Over Big Data

Module 6: Case Study - CTA

Robert Grossman
Open Data Group & Univ. of Chicago
Collin Bennett
Open Data Group

October 23, 2012
Chicago Transit Authority

- The Chicago Transit Authority provides service to Chicago and surrounding suburbs
- CTA operates 24 hours each day and on an average weekday provides 1.7 million rides on buses and trains
- It has 1,800 buses, operating over 140 routes, along 2,230 route miles
- Buses provide approximately 1 million rides day and serve more than 12,000 posted bus stops
CTA Data

• The City of Chicago has a public data portal
  • https://data.cityofchicago.org/

• Some of available categories are:
  • Administration & Finance
  • Community & Economic Development
  • Education
  • Facilities & Geographic Boundaries
  • Transportation
Moving to Hadoop

• For an example, we model ridership by bus route
  Data set: CTA - Ridership - Bus Routes - Daily Totals by Route

• MapReduce
  — Map Input:
    • Route number, Day type, Number of riders
  — Map Output / Reduce Input:
    • Key \( \text{t value} = \text{Route} \ \text{t day, date, riders} \)
  — Reduce Output:
    • Model for each route
Example Data Set

- Data goes back to 2001
  - Route Number
  - Date
  - Day of Week Type
  - Number of riders

- Demo size:
  - 11 MB
  - 534,208 records

- Saved as csv file in HDFS
Workflow

HDFS -> MAP -> SHUFFLE / SORT -> REDUCE

- **Record** -> **key, value**, **key, value**, **key, value**, **key, value**, **key, value**
- **key x**: **value**, **value**
- **key y**: **value**, **value**
- **key z**: **value**, **value**

1 key == 1 segment
Data has multiple keys
Model has multiple segments

Values used to compute model segment
Segment Identifier
Mapper

- Using Hadoop’s Streaming Interface, we can use a language appropriate for each step
- The mapper needs to read the records quickly
- We use Python

```python
#!/usr/bin/env python

import sys
import time

if __name__ == "__main__":
    for line in sys.stdin.readlines():
        route, date, daytype, rides = line.rstrip().split("","")
        weekday = time.strftime("%A", time.strptime(date, "%m/%d/%Y"))
        sys.stdout.write("%s-%s\t%s,%s\n" % (route, weekday, date, rides))
```
Reducer

• For this example, we calculate a simple statistic:
  – Gaussian Distribution of riders for route, day of week pairs described by the mean and variance
• We use the Baseline Model in PMML to describe this.
• The Baseline model is often useful for Change Detection.
  • Do the records matching a segment fit the expected distribution describing the segment
Describing a Segment

• Baseline Model

```xml
<BaselineModel functionName="regression">
  <MiningSchema>
    <MiningField usageType="active"
                 name="rides" />
  </MiningSchema>
  <TestDistributions field="rides"
                     testStatistic="zValue">
    <Baseline>
      <GaussianDistribution
            variance="64308.1625973"
            mean="1083.88950936">
      </Baseline>
    </GaussianDistribution>
  </TestDistributions>
</BaselineModel>
```
Building PMML in the Reducer

• You can embed a PMML engine into the reducer if you are using
  – Java
  – R
  – Python

• We present a light-weight solution which builds valid models, but also allows you to construct invalid ones
Using a Valid Model Template

```xml
# the segment is validated as PMML on load
segment = load(""
  <Segment>
    <SimplePredicate field="segment" operator="equal" value="zero"/>
    <BaselineModel functionName="regression">
      <MiningSchema>
        <MiningField usageType="active" name="rides"/>
      </MiningSchema>
      <TestDistributions field="rides" testStatistic="zValue">
        <Baseline>
          <GaussianDistribution mean="0" variance="1"/>
        </Baseline>
      </TestDistributions>
    </BaselineModel>
  </Segment>
"", pmml.PMML)
```
Template in R

• Similar but a little less elegant in R

```r
templateString <- addNamespaceAttrub(paste(
  "<Segment>",
  "  <SimplePredicate field="segment" operator="equal" value="zero"/>",
  "  <BaselineModel functionName="regression">
    <MiningSchema>
      <MiningField usageType="active" name="rides" />
    </MiningSchema>
    <TestDistributions field="rides" testStatistic="zValue">
      <Baseline>
        <GaussianDistribution mean="0" variance="1" />
      </Baseline>
    </TestDistributions>
  </BaselineModel>
"<Segment>",
  sep="\n"))
```
Populating the Template (Python)

- This is done in an XML / XPATH way, rather than as PMML

```python
def doLast(v):
    if v["count"] > 1:
        variance = v["varn"] * v["count"] / (v["count"] - 1.)
    else:
        variance = v["varn"]

    v["gaussianDistribution"]["mean"] = v["mean"]
    v["gaussianDistribution"]["variance"] = variance
    v["partialSum"]).attrib =
        {"COUNT": v["count"], "RUNMEAN": v["mean"], "RUNSN": v["varn"]})

    print v["segment"].xml()
```
Populating the Template (R)

doLast <- function(v) {
  # make a data frame out of the temporary table
dataframe <- do.call("rbind", v[['rows']])
colnames(dataframe) <- c("date", "rides")

  # pull a column out and call the function on it
rides <- unlist(dataframe[,"rides"])
result <- ewup2(rides, alpha)

  count <- result[['count']][[length(rides)]]
mean <- result[['mean']][[length(rides)]]
variance <- result[['variance']][[length(rides)]]
varn <- result[['varn']][[length(rides)]]

  # put the current mean and variance into the GaussianDistribution
xmlAttrs(v[['gaussianDistribution']]) <- c(mean=mean, variance=variance)

  # put the final state of the counters into the X-ODG-PartialSums
xmlAttrs(v[['partialSum']]) <- c(COUNT=count, RUNMEAN=mean, RUNSN=varn)

  # unzip R's lips just long enough to write our output
sink()
  print(v[['segment']])
sink("/dev/null")
}
Reducer Trade Offs

• Standard trade offs apply
  – Holding events in the reducer (easier, RAM) vs.
  – Computing running statistics (more coding, CPU)

• We present two techniques using R
  – Holding all events for a given key in a dataframe and then calculating the statistic when a key transition is identified
  – Holding the minimum data to calculate running statistics. Computed as each event is seen
Two Approaches

• Holding events requires extra memory and may get slow as the number of events shuffled to the reducer for a given keys grows large
  – But, it there are a lot of statistical and data mining methods available in R that operate on a dataframe

• Tracking running statistics requires more coding
  – But may be the only way to process a batch without spilling to disk
Dataframe

- Accumulate each mapped record
- For each event, add it to a temporary table

```r
doAny <- function(v, date, rides) {
    # add a column to the temporary table
    v[["rows"]][[length(v[["rows"]]) + 1]] <- list(date, rides)
}
```

- Create the dataframe when we have them all
When you have seen all of the, process and and compute

doLast <- function(v) {
  # make a data frame out of the temporary table
  dataframe <- do.call("rbind", v["rows"])
  colnames(dataframe) <- c("date", "rides")

  # pull a column out and call the function on it
  rides <- unlist(dataframe[, "rides"])
  result <- ewup2(rides, alpha)

  count <- result["count"][length(rides)]
  mean <- result["mean"][length(rides)]
  variance <- result["variance"][length(rides)]
  varn <- result["varn"][length(rides)]
}

Dataframe continued
Running Totals

• Update statistics as each event is encountered

```r
doNonFirst <- function(v, x) {
  # update the counters
  v[['count']] <- v[['count']] + 1
  diff <- v[['prevx']] - v[['mean']]
  incr <- alpha * diff
  v[['mean']] <- v[['mean']] + incr
  v[['varn']] <- (1.0 - alpha) *
      (v[['varn']] + incr*diff)
  v[['prevx']] <- x
}
```
Running Totals continued

• Update statistics as each event is encountered

```r
doLast <- function(v) {
  if (v[['count']] >= 2) {
    # when N >= 2, correct for 1/(N - 1)
    variance <- v[['varn']] / 
               (v[['count']] - 1.0)
  } else {
    variance <- v[['varn']] 
  }
}
```
Results

- As new data are seen, models can be updated
- New routes added as new segments
- Model is not bound to Hadoop
EDA

• Write plots to HDFS as SVG files (text, xml)
• Graphical view of the reduce calculation or statistic

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-/W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg style="stroke-linejoin:miter; stroke:black; stroke-width:2.5; text-anchor:middle; fill:none" xmlns="http://www.w3.org/2000/svg" font-family="Helvetica, Arial, FreeSans, Sans, sans, sans-serif" width="2000px" height="500px" version="1.1" xmlns:xlink="http://www.w3.org/1999/xlink" viewBox="0 0 2000 500">
  <defs>
    <clipPath id="Legend_0_clip">
      <rect x="1601.2" y="25" width="298.8" height="215" />
    </clipPath>
    <clipPath id="TimeSeries_0_clip">
      <rect x="240" y="25" width="1660" height="435" />
    </clipPath>
    <clipPath id="TimeSeries_10_clip">
      <rect x="240" y="25" width="1660" height="435" />
    </clipPath>
    <clipPath id="TimeSeries_11_clip">
      <rect x="240" y="25" width="1660" height="435" />
    </clipPath>
  </defs>
</svg>
```
Questions?

For the most current version of these slides, please see tutorials.opendatagroup.com
About Robert Grossman

Robert Grossman is the Founder and a Partner of Open Data Group, which specializes in building predictive models over big data. He is the Director of Informatics at the Institute for Genomics and Systems Biology (IGSB) and a faculty member in the Computation Institute (CI) at the University of Chicago. His areas of research include: big data, predictive analytics, bioinformatics, data intensive computing and analytic infrastructure. He has led the development of new open source software tools for analyzing big data, cloud computing, data mining, distributed computing and high performance networking. Prior to starting Open Data Group, he founded Magnify, Inc. in 1996, which provides data mining solutions to the insurance industry. Grossman was Magnify’s CEO until 2001 and its Chairman until it was sold to ChoicePoint in 2005. He blogs about big data, data science, and data engineering at rgrossman.com.
About Robert Grossman (cont’d)

• You can find more information about big data and related areas on my blog:
  
  rgrossman.com.

• Some of my technical papers are available online at rgrossman.com

• I just finished a book for the non-technical reader called *The Structure of Digital Computing: From Mainframes to Big Data*, which is available from Amazon.
About Collin Bennett

Collin Bennett is a principal at Open Data Group. In three and a half years with the company, Collin has worked on the open source Augustus scoring engine and a cloud-based environment for rapid analytic prototyping called RAP. Additionally, he has released open source projects for the Open Cloud Consortium. One of these, MalGen, has been used to benchmark several parallel computation frameworks. Previously, he led software development for the Product Development Team at Acquity Group, an IT consulting firm head-quartered in Chicago. He also worked at startups Orbitz (when it was still was one) and Business Logic Corporation. He has co-authored papers on Weyl tensors, large data clouds, and high performance wide area cloud testbeds. He holds degrees in English, Mathematics and Computer Science.
About Open Data

• Open Data began operations in 2001 and has built predictive models for companies for over ten years

• Open Data provides management consulting, outsourced analytic services, & analytic staffing

• For more information
  • www.opendatagroup.com
  • info@opendatagroup.com
About Open Data Group (cont’d)

Open Data Group has been building predictive models over big data for its clients over ten years. Open Data Group is one of the pioneers using technologies such as Hadoop and NoSQL databases so that companies can build predictive models efficiently over all of their data. Open Data has also participated in the development of the Predictive Model Markup Language (PMML) so that companies can more quickly deploy analytic models into their operational systems.

For the past several years, Open Data has also helped companies build predictive models and score data in real time using elastic clouds, such as provided with Amazon’s AWS.

Open Data’s consulting philosophy is nicely summed up by Marvin Bower, one of McKinsey & Company’s founders. Bower led his firm using three rules: (1) put the interests of the client ahead of revenues; (2) tell the truth and don’t be afraid to challenge a client’s opinion; and (3) only agree to perform work that is necessary and something [you] can do well.

www.opendatagroup.com