Teaching Big Data Analytics to Business School MS Students

Ramesh Shankar
Uconn School of Business
IT Teaching Workshop 2019, Wharton
MSBAPM Curriculum

Business Analytics

Required courses (5):
- Business Process Modeling and Data Management (OPIM 5272)
- Statistics in Business Analytics (OPIM 5603)
- Predictive Modeling (OPIM 5604)
- Business Decision Modeling (OPIM 5641)
- Data Mining and Business Intelligence (OPIM 5671)

Electives

- Visual Analytics (OPIM 5501)
- Big Data Analytics with Hadoop (OPIM 5502)
- Adaptive Business Intelligence (OPIM 5504)
- Analytical Consulting for Financial Services (OPIM 5505)
- Agile Project Management (OPIM 5507)
- Healthcare Analytics and Research Methods (OPIM 5508)
- Introduction to Deep Learning (OPIM 5509)
- Web Analytics (OPIM 5510)
- Survival Analysis using SAS BASE (OPIM 5511)
- Data Science using Python (OPIM 5512)

Project Management

Required courses (4):
- Introduction to Project Management (OPIM 5270)
- Project Leadership and Communications (MGMT 5620)
- Project Risk and Cost Management (OPIM 5668)
- Advanced Business Analytics and Project Management (OPIM 5770)

MSBAPM Capstone Project

Quick Facts

<table>
<thead>
<tr>
<th>Locations</th>
<th>Hartford or Stamford, Connecticut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semesters</td>
<td>Fall, Spring, or Summer</td>
</tr>
<tr>
<td>Format</td>
<td>Full- or Part-time</td>
</tr>
<tr>
<td>Credits</td>
<td>37-credits (Curriculum)</td>
</tr>
</tbody>
</table>
Hadoop books

Source: David Tilson, IT Teaching Workshop 2018
Hadoop resources

QuickStarts for CDH 5.13
Virtualized clusters for easy installation on your desktop.

Cloudera QuickStart VMs (single-node cluster) make it easy to quickly get hands-on with CDH for testing, demo, and self-learning purposes, and include Cloudera Manager for managing your cluster. Cloudera QuickStart VM also includes a tutorial, sample data, and scripts for getting started.

Cloudera University
Hone your big data tech skills with the world's leading experts through Cloudera University — the industry's only truly dynamic training curriculum that's updated to keep pace with innovation.

FIND TRAINING  GET CERTIFIED  RETURNING STUDENTS
Cloudera VM

Enabling virtualization
AWS EMR (Elastic MapReduce) Cluster

Using a cluster is not for the faint-hearted

**Estimated cost**
- $6k cluster time (Spark was most expensive part)
- $2k admin time
- $10k consulting time (one-off)

Asking AWS for $5k credit (~50 students) and they are considering more turn-key solution (no promises yet)

>50% cost reduction by active management (off at night, reset). Could be less than $50/student at 50 students for 10 weeks

**AWS provide good support (solution architects)**
- But they were learning too
- Multi-user different and complex across AWS, Linux, HDFS, Pig, Hive, S3, Hue, Spark, and Zeppelin
- Will use us as case study in multi-tenancy cluster operation

---

This book was very helpful in learning AWS terminology

Amazon provides lots of guidance


Source: David Tilson, IT Teaching Workshop 2018
Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application, etc.) provided by AWS, our user community, or the AWS Marketplace. Or, you can select one of your own AMIs.

Search for an AMI: v4-ubuntu-hadoop-rs

- Quick Start (0)
- My AMIs (2)
- AWS Marketplace (287)

Community AMIs (1)

- Operating system
  - Amazon Linux
  - CentOS
  - Debian
  - Fedora
  - Ubuntu

The following results for "v4-ubuntu"

1 results in My AMIs
My AMIs are AMIs owned by you or shared with you.

287 results in AWS Marketplace
AWS Marketplace provides partnered software.

Select v4-ubuntu-hadoop-rs - ami-1234567890

Root device type: ebs
Virtualization type: t1
64-bit (x86)
AWS EC2:
Topics covered

• Linux
• Hadoop Distributed File System
• Apache Sqoop
  • Extract data from RDBMS, into HDFS
• Apache Pig
  • Extract, Transform, Load (ETL) on data obtained via Sqoop
  • Schema on read, no permanent schema, flat files
• Apache Hive
  • Hadoop Data Warehousing Tool
  • Schema on read, permanent schema required, flat files
• MapReduce – conceptual overview
• Spark
  • In-memory Analytics
• Recommender Systems
  • Illustrates Spark
HDFS

```
[training@localhost mydata]$ hadoop fs -mkdir /test
[training@localhost mydata]$ hadoop fs -mkdir /test/test1
[training@localhost mydata]$ hadoop fs -put products.txt /test/test1/
[training@localhost mydata]$ hadoop fs -ls /test/test1
Found 1 items
-rw-r--r-- 1 training supergroup 63 2017-10-17 12:25 /test/test1/products.txt
```

```
[training@localhost mydata]$ hadoop fs -rm /test/test1
rm: `/test/test1': Is a directory
```

```
[training@localhost mydata]$ hadoop fs -rmdir /test/test1
rmdir: `/test/test1': Directory is not empty
```

```
[training@localhost mydata]$ hadoop fs -rm -r /test/test1
Deleted /test/test1
```

```
[training@localhost mydata]$ hadoop fs -ls /test
```

```
$ hadoop fs -ls /  
Found 6 items
drwxr-xr-x  - ubuntu supergroup 2019-03-11 20:28 /linkage
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
$ sqoop import \\
--connect jdbc:mysql://localhost/sakila \\
--username ubuntu --password training \\
--warehouse-dir /userdata \\
--table actor
Pig

1 listings = LOAD '/mydata/class3/listings.txt' AS
2 ( 
3 listing_id:int, date_listed:chararray, 
4 list_price:float, sq_feet:int, 
5 address:chararray, town:chararray 
6 );
7 bytown = GROUP listings BY town;
8 DESCRIBE bytown;
9 --optional step:
10 --byproduct = LIMIT byproduct 5;
11 --Top 2 most expensive homes per town
12 top_homes = FOREACH bytown { 
13   sorted = ORDER listings BY 
14     list_price DESC; 
15   most_expensive = LIMIT sorted 2; 
16   GENERATE group, most_expensive; 
17 };
18 DESCRIBE top_homes;
19 DUMP top_homes;

- Load data
- Group by town
- In each group, sort by list_price
- Limit to top 2 most expensive homes
- Generate new (group) record in top_homes table
hive> SELECT * FROM homes;

OK

<table>
<thead>
<tr>
<th>listing_id</th>
<th>list_price</th>
<th>sqft</th>
<th>realtor_id</th>
<th>town</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>146000.0</td>
<td>1750</td>
<td>25</td>
<td>Storrs</td>
</tr>
<tr>
<td>2</td>
<td>235000.0</td>
<td>2100</td>
<td>17</td>
<td>Storrs</td>
</tr>
<tr>
<td>3</td>
<td>101000.0</td>
<td>1550</td>
<td>53</td>
<td>Hartford</td>
</tr>
<tr>
<td>4</td>
<td>376000.0</td>
<td>2900</td>
<td>17</td>
<td>Storrs</td>
</tr>
<tr>
<td>5</td>
<td>291000.0</td>
<td>2400</td>
<td>17</td>
<td>Hartford</td>
</tr>
<tr>
<td>6</td>
<td>409000.0</td>
<td>3500</td>
<td>25</td>
<td>Stamford</td>
</tr>
</tbody>
</table>

Time taken: 0.129 seconds

hive> SELECT * FROM realtors;

OK

<table>
<thead>
<tr>
<th>realtor_id</th>
<th>realtor</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Alec Baldwin</td>
</tr>
<tr>
<td>25</td>
<td>Al Pacino</td>
</tr>
<tr>
<td>53</td>
<td>Kevin Spacey</td>
</tr>
</tbody>
</table>

Time taken: 0.484 seconds

hive> SELECT r.realtor, h.list_price, h.sqft, h.town
   > FROM realtors r JOIN homes h
   > ON r.realtor_id = h.realtor_id
   >;

<table>
<thead>
<tr>
<th>realtor</th>
<th>list_price</th>
<th>sqft</th>
<th>town</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alec Baldwin</td>
<td>235000.0</td>
<td>2100</td>
<td>Storrs</td>
</tr>
<tr>
<td>Alec Baldwin</td>
<td>376000.0</td>
<td>2900</td>
<td>Storrs</td>
</tr>
<tr>
<td>Alec Baldwin</td>
<td>291000.0</td>
<td>2400</td>
<td>Hartford</td>
</tr>
<tr>
<td>Al Pacino</td>
<td>146000.0</td>
<td>1750</td>
<td>Storrs</td>
</tr>
<tr>
<td>Al Pacino</td>
<td>409000.0</td>
<td>3500</td>
<td>Stamford</td>
</tr>
<tr>
<td>Kevin Spacey</td>
<td>101000.0</td>
<td>1550</td>
<td>Hartford</td>
</tr>
</tbody>
</table>

Time taken: 81.428 seconds
MapReduce

Map tasks

File1.txt
- C
  - (cat, 2)
  - (drank, 1)
  - (milk, 1)

File2.txt
- C
  - (dog, 1)
  - (drank, 1)
  - (milk, 1)

File3.txt
- C
  - (dog, 1)
  - (chased, 1)
  - (cat, 1)

Key value pairs

Values grouped by key

Value combined by key

M
- (cat, 2)
- (drank, 1)
- (milk, 1)

C
- (cat, 2, 1)
- (drank, 1, 1)
- (milk, 1, 1)

C
- (dog, 1)
- (chased, 1)
- (cat, 1)

C
- (cat, 3)
- (drank, 2)
- (milk, 2)
- (dog, 2)
- (chased, 1)

Map step – in child computers

Aggregation step – in master computer

Reduce step – in child computers
(userid artistid playcount)

(user_artist_data.txt)

1 1000002  1 55
2 1000002  1000006  33
3 1000002  1000007  8
4 1000002  1000009  144
5 1000002  1000010  314
6 1000002  1000013  8
7 1000002  1000014  42
8 1000002  1000017  69
9 1000002  1000024  329

(artistid artist_name)

(artist_data.txt)

1 1 Portishead
2 100 Phoenix
3 1000006 Phil Collins Big Band
4 1000007 The Phil Collins Big Band
5 1000009 A Perfect Circle
6 1000010 Aerosmith
7 1000013 MC Hawking
8 1000014 Pantera
9 1000017 Judas Priest

(badid goodid)

(artist_alias.txt)

1 1000434  1000518
2 1021484  1234336
3 1014609  1014609
4 1000287  1239413
5 1004729  1003612
6 1006586  1021625
7 1008128  1019469
8 1002081  1013150
9 1939  6785079
Spark – recommender system (ALS)

```scala
artistByID.filter { case (id, name) =>
  existingProducts.contains(id)
}.values().foreach(println)
```

Some of the artists this person listens to:

- Sonny Rollins
- Thelonious Monk
- Sublime
- Weather Report
- Bob Dylan
- Pink Floyd
- Nine Inch Nails
- Otis Redding
- Stevie Wonder

Make 5 recommendations to this user:

```scala
scala> val recommendedProductIDs = recommendations.map(_.product).toSet
```

```scala
scala> artistByID.filter { case (id, name) =>
  | recommendedProductIDs.contains(id)
  | }.values().foreach(println)
```

- Lee Ritenour
- 60ft Dolls
- Belly
- Ella Fitzgerald

```scala
User ID, Artist ID, estimated rating
Rating(1000002, 1003433, 1.1815765349999169)
Rating(1000002, 719, 1.1266192051236328)
Rating(1000002, 1001172, 1.1254197498636407)
Rating(1000002, 1000840, 1.097738376342873)
Rating(1000002, 1034635, 1.0651848760166387)
```