

Building Custom Machine Learning Algorithms with Apache SystemML

Fred Reiss

Chief Architect, IBM Spark Technology Center
Member, IBM Academy of Technology



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Roadmap

- What is Apache SystemML?
- Demo!
- How to get SystemML



What is Apache SystemML?



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Origins of the SystemML Project

You are
here.

2015

2016



SPARK SUMMIT 2016

2011

2012

2013

2014



SPARK SUMMIT 2016

2007-2008: Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

2009: We form a dedicated team for scalable ML

2009-2010: Through engagements with customers, we observe how data scientists create **ML solutions**.

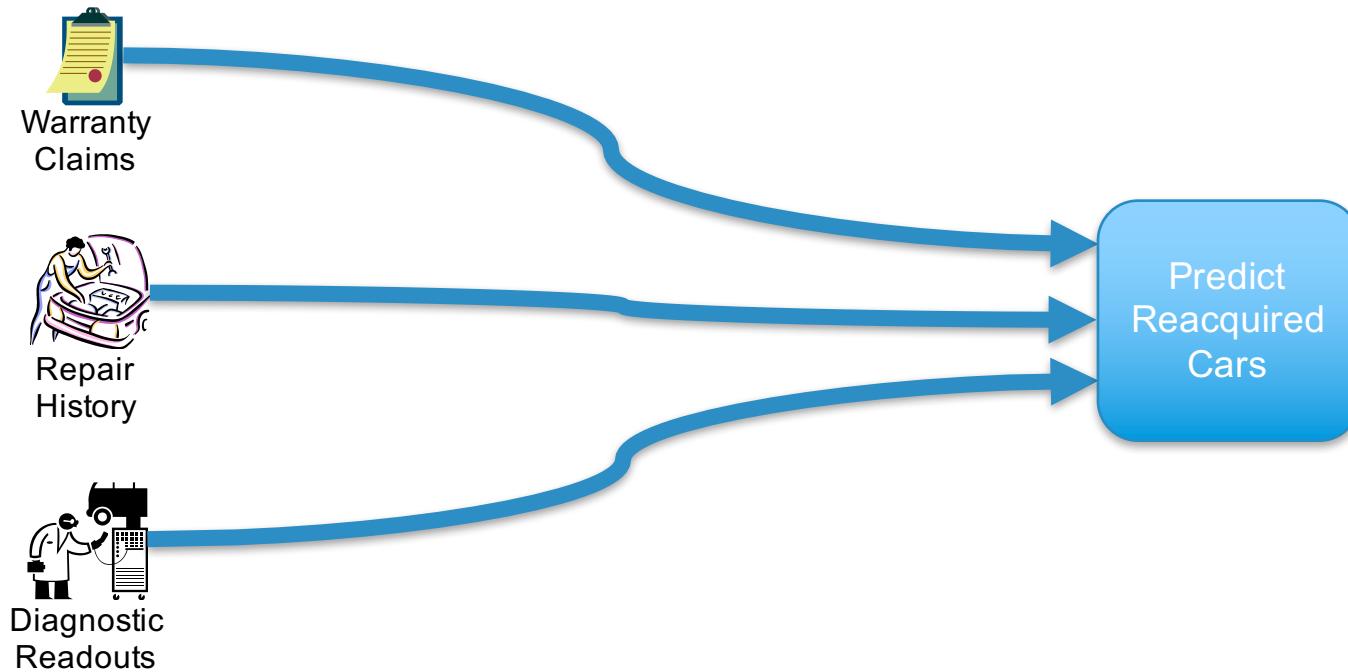
2007

2008

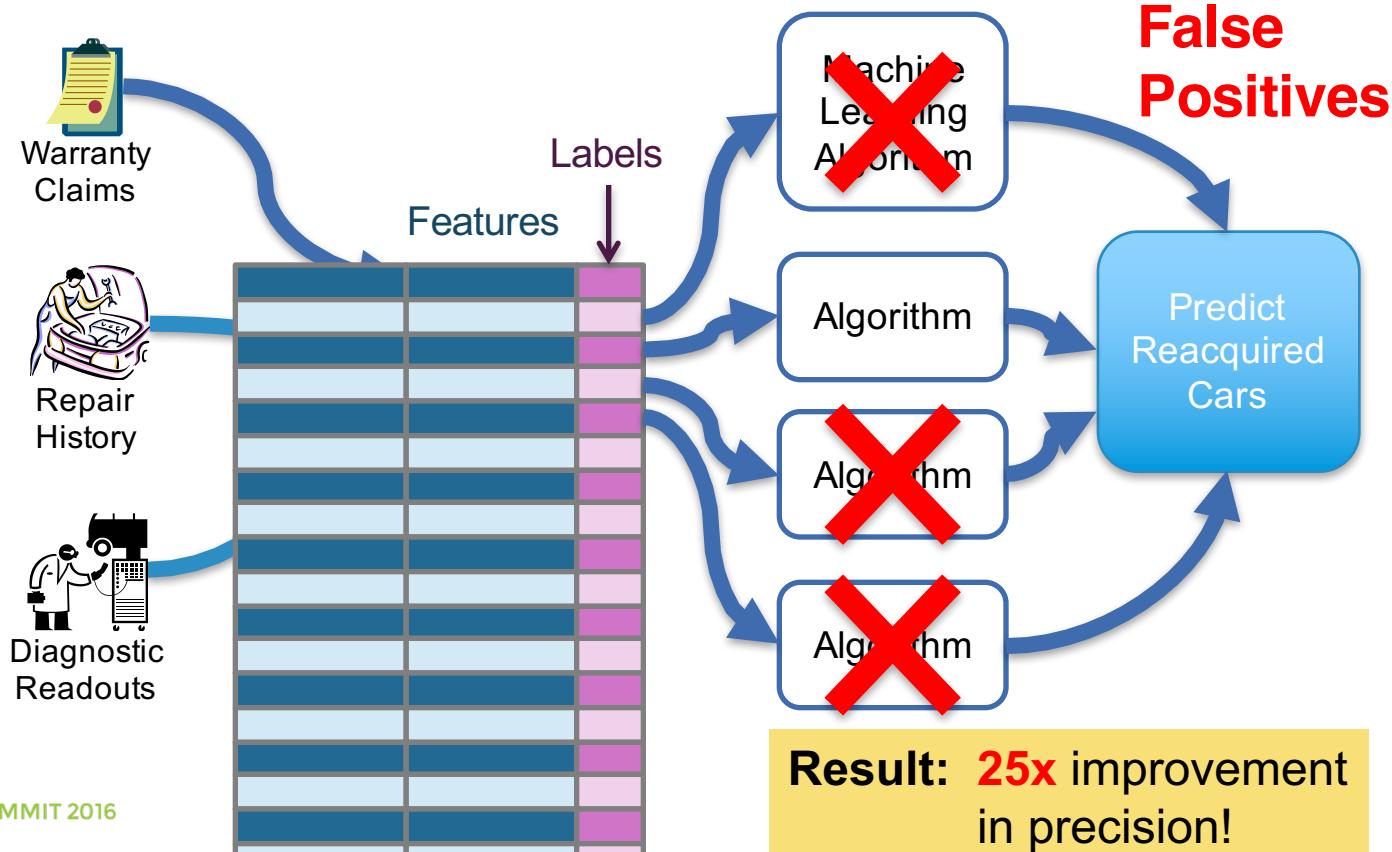
2009

2010

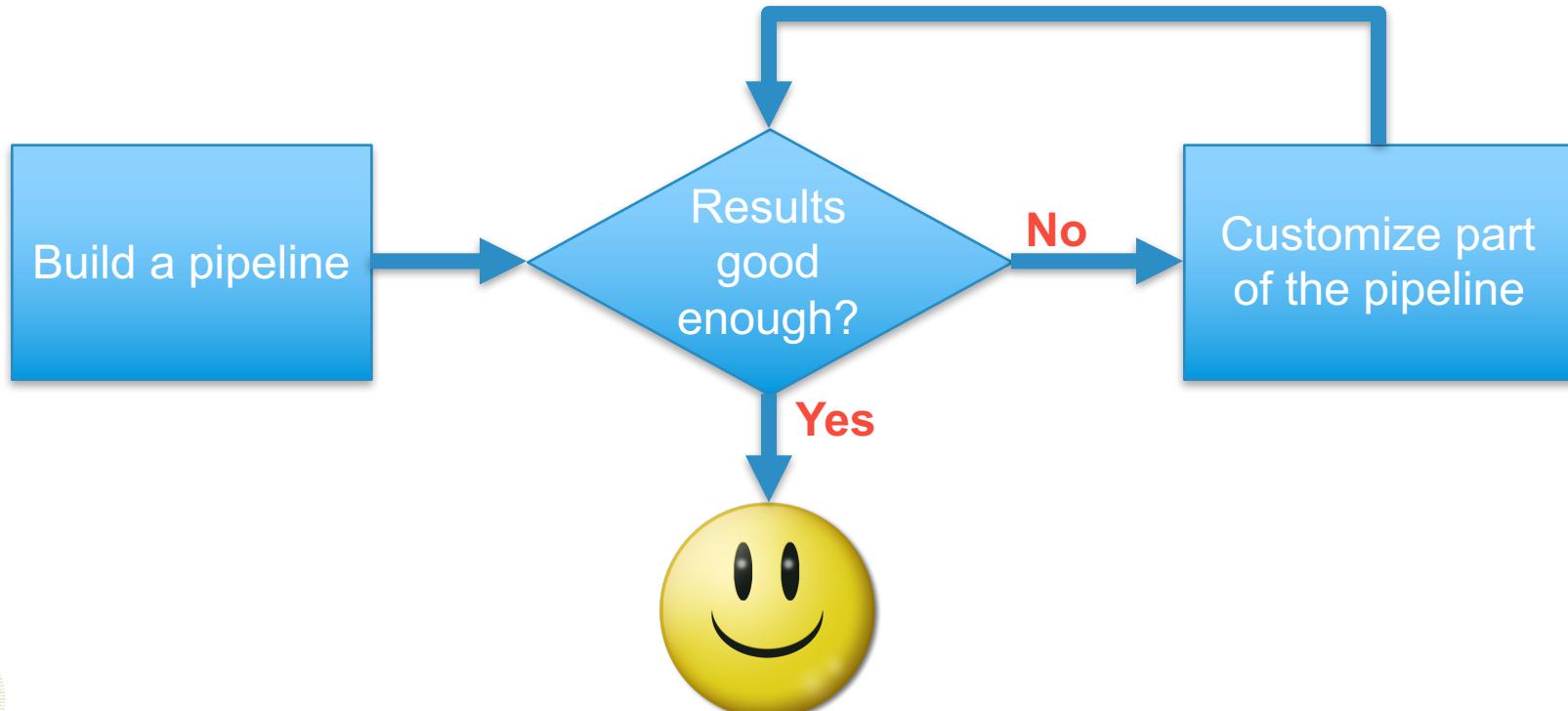
Case Study: An Auto Manufacturer



Case Study: An Auto Manufacturer

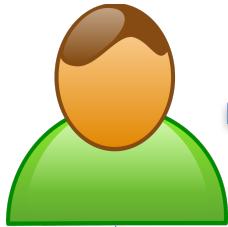


The Iterative Development Process



State-of-the-Art: Small Data

Data
Scientist



R or
Python

```
4 X = read ('X'); # explanatory variables
5 y = read ('y'); # predicted variables
6
7 n = nrow (X);
8 m = ncol (X);
9
10 # Rescale the columns in X if needed
11 scale_lambda = matrix (1, rows = 1, cols = m);
12 lambda = t(scale_lambda); #reg;
13
14 # Construct the covariance matrix of error
15 A = t(X) %*% lambda;
16 b = t(X) %*% y;
17
18 beta = solve (A, b);
19 ...
20 write (beta, "B");
```

Data

Weather station	Average temperature (°C)	Min	Max	Daily rainfall (mm)	Mean	Max
RYGGE	-10.3	7.5	23.8	—	2.0	34.8
NESVYEN - TØDDEK	-16.8	4.8	21.8	—	1.2	35.9
TØRUGGEN FYR	-4.0	8.5	21.8	—	2.1	35.9
SØLA	-14.8	5.2	21.8	—	2.5	35.3
GÅRDERMOEN	-15.4	5.4	19.4	—	2.0	35.5
BERGEN - FLORIDA	-5.3	19.3	26.1	—	8.4	150.5
LEIRDAL - MØLDO	—	—	—	—	1.8	38.1
TØRUGGEN	—	—	—	—	2.6	34.3
VERNESS	—	—	—	—	2.5	27.0
RESA - RAUGEDALEN	8.2	5.7	22.3	—	1.9	36.6
BODO VI	8.2	5.7	22.3	—	4.0	30.5
TØNSBERG	4.4	1.1	21.1	—	2.5	35.3
KAUTOKERINO	-29.8	-0.5	21.1	—	1.5	18.8
NY-ÅLESUND	-24.4	-3.4	13.2	—	0.9	29.1
JAN MAYEN	-11.8	0.6	11.1	—	2.0	23.3



Personal
Computer

Results

1	AAPL	30/05/2008	182.75	188.75
2	AAPL	08/06/2008	188.6	185.64
3	AAPL	13/06/2008	184.79	172.37
4	AAPL	20/06/2008	171.3	175.27
5	AAPL	27/06/2008	173.14	173.16
6	AAPL	03/07/2008	173.29	172.12
7	AAPL	10/07/2008	172.11	172.58
8	AAPL	17/07/2008	172.21	172.58
9	AAPL	01/08/2008	162.34	162.12
10	AAPL	08/08/2008	158.56	165.66
11	AAPL	15/08/2008	175.97	175.34
12	AAPL	22/08/2008	175.57	176.79
13	AAPL	29/08/2008	178.15	169.53

State-of-the-Art: Big Data

Data
Scientist



```
4 X = read (x); # explanatory variables
5 y = read (y); # predicted variables
6
7 n = nrow (X);
8 m = ncol (X);
9
10 # Rescale the columns of X if needed
11 scale_lambda = matrix (1, row = 1, cols = m);
12 lambda = t (scale_lambda) * diag (X);
13
14 # Construct and solve the OLS equations
15 A = t (X) %*% y;
16 b = t (X) %*% y;
17
18 beta = solve (A, b);
19 ...
20 write (beta, <B>);
```

Systems
Programmer



R or
Python

```
object BoxPack extends InventoryItem {
  case class MoreInventoryItem(val needsInventoryItem: String)
  object MoreInventoryItem {
    def unapply(item: MoreInventoryItem): Option[InventoryItem] = item match {
      case MoreInventoryItem(item) => Some(item)
    }
  }
}
val gr = new Generator
val root = gr.rootBox
val box = BoxPack("box")
box.setNeedsInventoryItem("box")
box.setInventory(gr.inventory.filter(_.isBuffer))
val library = BoxPack("library")
val ladder = BoxPack("ladder")
ladder.setNeedsInventoryItem("box")
ladder.setInventory(gr.inventory.filter(_.isBox))
case class TransitionPlan(Places: Map[String, Seq[InventoryItem]], val: Int)
val transitionPlans = [
  TransitionPlan(Map("box" -> Set(box)), 1),
  TransitionPlan(Map("box" -> Set(library, ladder)), 2),
  TransitionPlan(Map("library" -> Set(library)), 3),
  TransitionPlan(Map("ladder" -> Set(ladder)), 4)
]
```

Scala

AAPL	30/05/2008	182.75	188.75
AAPL	06/06/2008	188.6	185.64
AAPL	13/06/2008	184.79	172.37
AAPL	20/06/2008	171.3	175.27
AAPL	27/06/2008	174.74	170.09
AAPL	03/07/2008	170.19	170.12
AAPL	10/07/2008	173.16	172.58
AAPL	17/07/2008	170.4	165.15
AAPL	25/07/2008	166.9	162.12
AAPL	01/08/2008	162.34	156.66
AAPL	08/08/2008	158.6	169.55
AAPL	15/08/2008	170.07	175.74
AAPL	22/08/2008	175.57	176.79
AAPL	29/08/2008	176.15	169.53

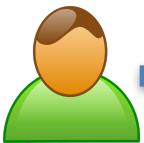
Results



Spark

State-of-the-Art: Big Data

Data
Scientist



Days or weeks per iteration
Errors while translating
algorithms



Spark



	Date	Open	Close	
22	AAPL	30/05/2008	182.75	188.75
23	AAPL	06/06/2008	188.6	185.64
24	AAPL	13/06/2008	184.79	172.37
25	AAPL	20/06/2008	171.3	175.27
26	AAPL	27/06/2008	174.74	170.09
27	AAPL	03/07/2008	170.19	170.12
28	AAPL	10/07/2008	173.16	172.58
29	AAPL	17/07/2008	170.4	165.15
30	AAPL	25/07/2008	166.9	162.12
31	AAPL	01/08/2008	162.34	156.66
32	AAPL	08/08/2008	156.6	169.55
33	AAPL	15/08/2008	170.07	175.74
34	AAPL	22/08/2008	175.57	176.79
35	AAPL	29/08/2008	176.15	169.53

Results

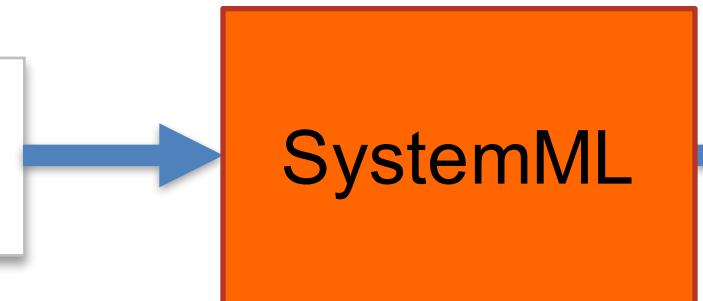
The SystemML Vision

Data
Scientist



R or
Python

```
4 X = read (4X); # explanatory variables
5 Y = read (6Y); # predicted variables
6
7 n = nrow (X);
8 m = ncol (X);
9
10 # Rescale the columns if needed
11 scale_lambda = matrix (1, row = 1, cols = m);
12 lambda = t (scale_lambda) * diag;
13
14 # Construct an active set regression equation
15 A = t (X) * lambda;
16 b = t (X) * y;
17
18 beta = solve (A, b);
19 ...
20 write (beta, <B>);
```



SystemML

Results

AAPL	30/05/2008	182.75	188.75
AAPL	06/06/2008	188.6	185.64
AAPL	13/06/2008	184.79	172.37
AAPL	20/06/2008	171.3	175.27
AAPL	27/06/2008	174.74	170.09
AAPL	03/07/2008	170.19	170.12
AAPL	10/07/2008	173.16	172.58
AAPL	17/07/2008	170.4	165.15
AAPL	25/07/2008	166.9	162.12
AAPL	01/08/2008	162.34	156.66
AAPL	08/08/2008	158.6	169.55
AAPL	15/08/2008	170.07	175.74
AAPL	22/08/2008	175.57	176.79
AAPL	29/08/2008	176.15	169.53

Spark

The SystemML Vision

Data
Scientist



R or
Python

```
4 X = read (6X); # explanatory variables
5 y = read (6Y); # predicted variables
6
7 n = nrow (X);
8 m = ncol (X);
9
10 # Rescale the columns in X if needed
11 scale_lambda = matrix (1, rows = 1, cols = m);
12 lambda = t (scale_lambda) * 4 * pi;
13
14 # Construct an inverse of the covariance matrix
15 A = t (X) * X + lambda * diag (m);
16 b = t (X) * y;
17
18 beta = solve (A, b);
19 ...
20 write (beta, 4B);
```



Fast iteration
Same answer

SystemML

Spark

Results

21	AAPL	30/05/2008	182.75	188.75
22	AAPL	08/06/2008	188.6	185.64
23	AAPL	13/06/2008	184.79	172.37
24	AAPL	20/06/2008	171.3	175.27
25	AAPL	27/06/2008	174.74	170.09
26	AAPL	03/07/2008	170.19	170.12
27	AAPL	10/07/2008	170.16	172.58
28	AAPL	17/07/2008	170.17	165.15
29	AAPL	25/07/2008	166.9	162.12
30	AAPL	01/08/2008	162.34	158.66
31	AAPL	08/08/2008	156.6	169.55
32	AAPL	15/08/2008	170.07	175.74
33	AAPL	22/08/2008	175.57	176.79
34	AAPL	29/08/2008	176.15	169.53



2007-2008: Multiple projects at IBM Research – Almaden involving machine learning on Hadoop.

2009: We form a dedicated team for scalable ML

2009-2010: Through engagements with customers, we observe how data scientists create machine learning algorithms.

2007

2008

2009

2010

Research

2011

2012

2013

2014



SPARK SUMMIT 2016

Apache SystemML

June 2015: IBM Announces open-source SystemML

November 2015: SystemML enters Apache incubation

June 2016: Second Apache release (0.10)

September 2015: Code available on Github

February 2016: First release (0.9) of Apache SystemML

2015

2016

SystemML at IBM Watson Health

- Built algorithms for predicting treatment outcomes
 - Substantial improvement in accuracy
- Moved from Hadoop MapReduce to Spark
 - SystemML supports both frameworks
 - **Exact same code**
 - **300X faster** on 1/40th as many nodes



SystemML at Cadent Technology



Cadent is a leading provider of TV advertising and data solutions, reaching over 140 million homes and trusted by the world's largest service providers.

“SystemML allows Cadent to implement advanced numerical programming methods in Apache Spark, empowering us to leverage specialized algorithms in our predictive analytics software.”

Michael Zargham
Chief Scientist



SPARK SUMMIT 2016

Demo!



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Demo Scenario

- **Application:** Targeted ads using demographic information tied to cookies
- **Problem:** The information is incomplete
- **Solution:** Estimate the missing values
 - Treat the problem as a **matrix completion** problem



Data

- The U.S. Census Public Use Microdata Sample (PUMS) data set for 2010
- 10% sample of the U.S. population
 - We'll use just California today
- Use this full data set to generate synthetic incomplete data

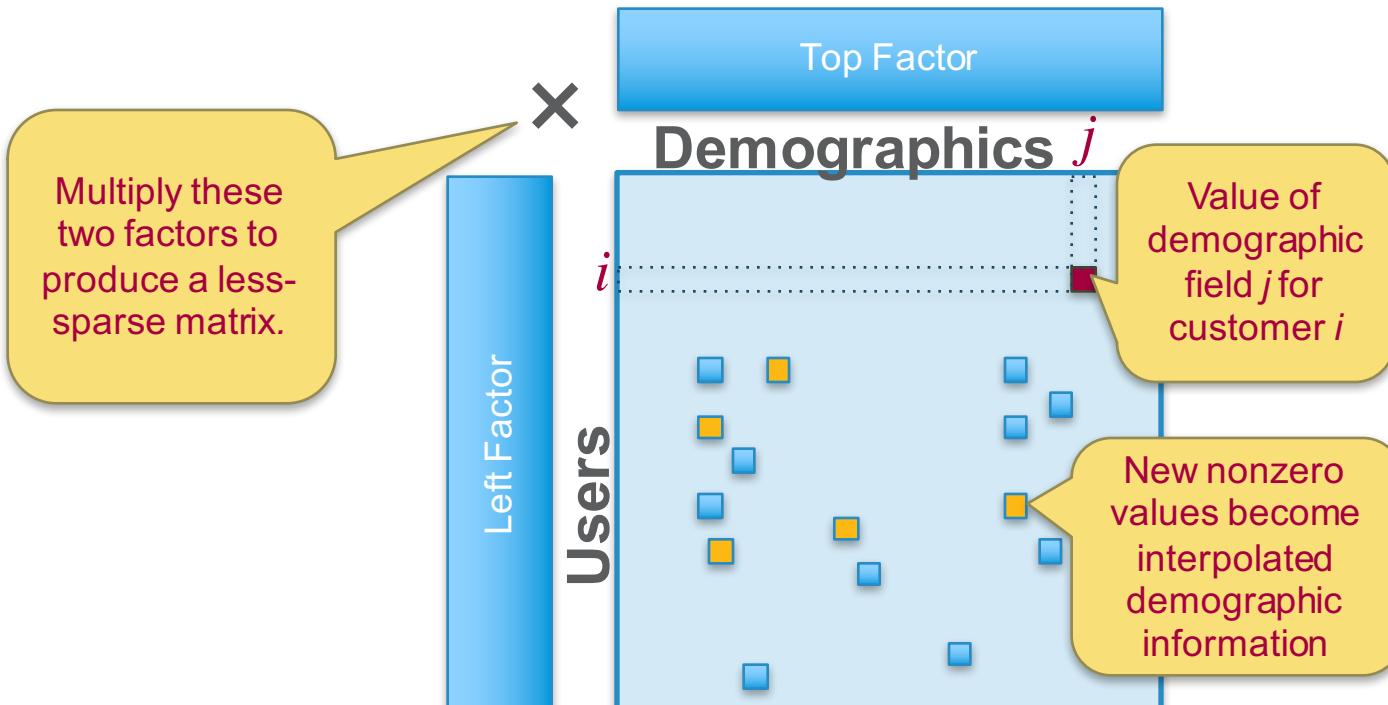


Demo Scenario

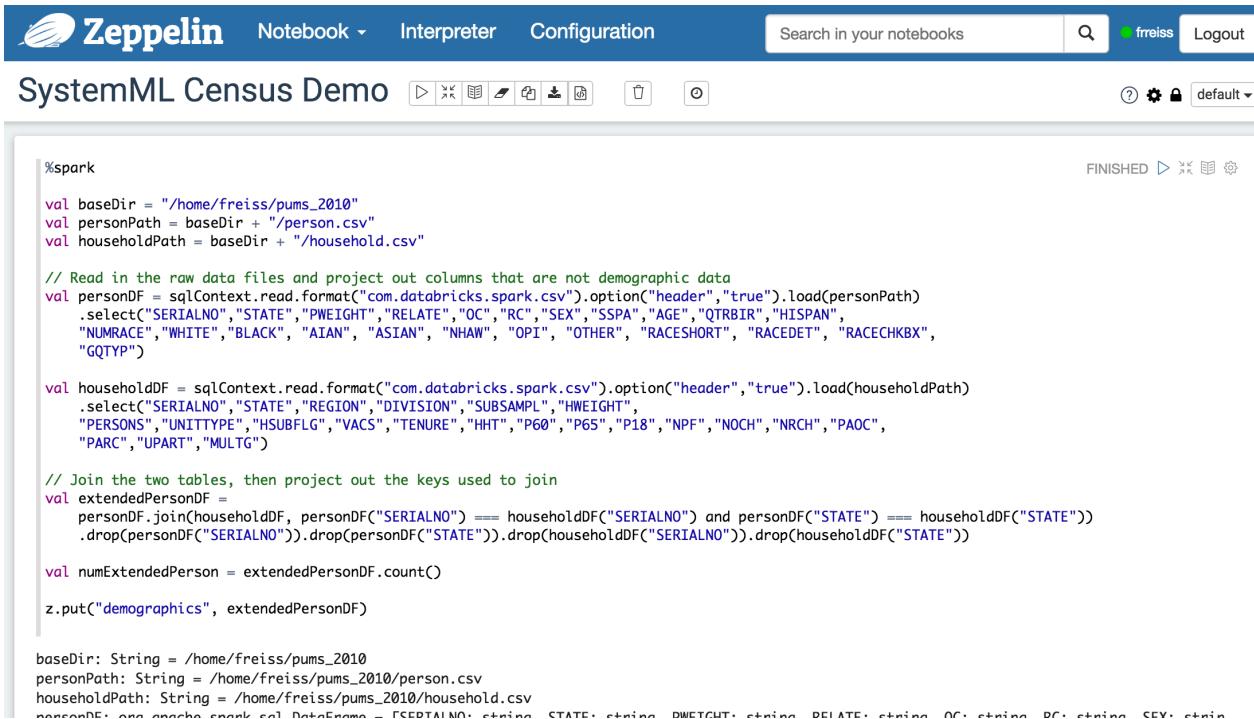
- **Application:** Identify products that are complementary (often purchased together)
- **Problem:** Customers are not currently buying the best complements at the same time
- **Solution:** Suggest new product pairings
 - Treat the problem as a **matrix completion** problem



Matrix Factorization



Demo Part 1: Data wrangling



The screenshot shows a Zeppelin notebook interface. At the top, there's a header bar with the Zeppelin logo, navigation links for Notebook, Interpreter, and Configuration, a search bar, and user information for 'frieiss'. Below the header is the title 'SystemML Census Demo'. The main area contains a code editor with SystemML code, a preview pane showing the results, and various toolbar icons.

```
%spark
val baseDir = "/home/frieiss/pums_2010"
val personPath = baseDir + "/person.csv"
val householdPath = baseDir + "/household.csv"

// Read in the raw data files and project out columns that are not demographic data
val personDF = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load(personPath)
  .select("SERIALNO", "STATE", "PWEIGHT", "RELATE", "OC", "RC", "SEX", "SSPA", "AGE", "QTRBIR", "HISPA",
  "NUMRACE", "WHITE", "BLACK", "AIAN", "ASIAN", "NHAW", "OPI", "OTHER", "RACESHORT", "RACEDET", "RACECHKBX",
  "GQTYP")

val householdDF = sqlContext.read.format("com.databricks.spark.csv").option("header", "true").load(householdPath)
  .select("SERIALNO", "STATE", "REGION", "DIVISION", "SUBSAMPL", "HWEIGHT",
  "PERSONS", "UNITYTYPE", "HSUBFLG", "VACS", "TENURE", "HHT", "P60", "P65", "P18", "NPF", "NOCH", "NRCH", "PAOC",
  "PARC", "UPART", "MULTG")

// Join the two tables, then project out the keys used to join
val extendedPersonDF =
  personDF.join(householdDF, personDF("SERIALNO") === householdDF("SERIALNO") and personDF("STATE") === householdDF("STATE"))
  .drop(personDF("SERIALNO")).drop(personDF("STATE")).drop(householdDF("SERIALNO")).drop(householdDF("STATE"))

val numExtendedPerson = extendedPersonDF.count()

z.put("demographics", extendedPersonDF)

baseDir: String = /home/frieiss/pums_2010
personPath: String = /home/frieiss/pums_2010/person.csv
householdPath: String = /home/frieiss/pums_2010/household.csv
personDF: org.apache.spark.sql.DataFrame = [SERIALNO: string, STATE: string, PWEIGHT: string, RELATE: string, OC: string, RC: string, SEX: string]
```



Demo Part 2: Custom algorithm

Algorithm Customizability

ML algorithms are expressed in an R-like or Python-like syntax that includes linear algebra primitives, statistical functions, and ML-specific constructs. This high-level language significantly increases the productivity of data scientists as it provides (1) full flexibility in expressing custom analytics, and (2) data independence from the underlying input formats and physical data representations. Automatic optimization according to data and cluster characteristics ensures both efficiency and scalability.

Poisson Nonnegative Matrix Factorization in SystemML's R-like Syntax

```
while (iter < max_iterations) {  
    iter = iter + 1;  
    H = (H * (t(W) %*% (V/(W%*%H)))) / t(colSums(W));  
    W = (W * ((V/(W%*%H)) %*% t(H))) / t(rowSums(H));  
    obj = as.scalar(colSums(W) %*% rowSums(H)) - sum(V * log(W%*%H));  
    print("iter=" + iter + " obj=" + obj);  
}
```



Key Points

- SystemML, Spark, and Zeppelin work together
- Linear algebra is great for data science
- Customization is important



How to get Apache SystemML



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

The Apache SystemML Web Site

<http://systemml.apache.org>

The screenshot shows the Apache SystemML website with a dark blue header. The header includes the Apache logo, the text "Apache SystemML", and a navigation bar with links for "Community", "GitHub", "Documentation", "Download", and "Apache". A dropdown menu under "Community" is open, showing "Get Involved" and "Who we are". Below the header, there's a large yellow speech bubble containing the text "Download the binary release!". Another yellow speech bubble contains "Contribute to the project!". To the right, two more yellow speech bubbles say "Browse the source!" and "Try out some tutorials!". At the bottom left, a blue button says "Get SystemML".

Apache SystemML

Community ▾

Get Involved

Who we are

GitHub Documentation Download Apache ▾

Download the binary release!

Contribute to the project!

Browse the source!

Try out some tutorials!

Get SystemML



SPARK SUMMIT 2016

THANK YOU.

Please try out Apache SystemML!

<http://systemml.apache.org>

Special thanks to Nakul Jindal and Mike Dusenberry for helping with the demo!



SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO