The Universal Java Matrix Package (UJMP)

Everything is a Matrix!

ICML/ MLOSS, Haifa, 2010-06-25

Outline

- Introduction
- Comparison of Matrix Libraries for Java
- Concepts for a Next-Generation Matrix Library
- Integration of Other Matrix Libraries
- Calculation Methods
- Matrix Annotation
- Automatic Entry Type Conversion
- Demo
- Summary and Discussion

Find out more at:
http://www.ujmp.org
Introduction

Why do we need yet another Java matrix library?

- Matrix computations essential in various fields of computer science (machine learning, data mining, etc.)
- Collaborative networks require large sparse adjacency matrices
- Increasing amount of data

But:

- No direct support for matrix algebra in JDK
- Matlab or Octave cannot always be used
- Other libraries have limitations: JAMA, Colt, MTJ, commons-math
Comparison of Matrix Libraries for Java

No single Java matrix library can fulfill all needs! We need a „universal“ matrix package...

<table>
<thead>
<tr>
<th>Feature</th>
<th>JAMA</th>
<th>Colt</th>
<th>MTJ</th>
<th>commons-math</th>
<th>UJMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>extendable</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>dense matrices</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>sparse matrices</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>2D matrices</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3D matrices</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>4D matrices</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>matrices &gt; 4D</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>&gt; 2^31 rows/columns</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>object entries</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>generic entries</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>matrices &gt; RAM</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>advanced operators</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>import/export filters</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Matlab/Octave/R interface</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>visualization methods</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Concepts for a Next-Generation Matrix Library

The actual implementation of a matrix becomes secondary!

**Matrix Interface** multi-dimensional, dense/sparse, $2^{63}$ rows/columns, various cell types

- size
- get/set cell
- plus, minus
- multiply, divide
- transpose
- min, max, mean
- variance, std
- sin, cos, tan
- select rows/cols
- get submatrix
- import/export
- visualization
  (list not complete)

**Abstract Matrix**

**Matrix Implementations**

- Data in Memory
  - double[][]
  - int[][]
  - String[][]

- Data on Disk
  - CSV, TXT

**Matrix Libraries**

- JAMA
- Colt
- ojAlgo

**Custom Function Implementations**

**Default Function Implementations**

**Java Libraries**

- EHCache

**Database Tables**

- Apache Derby
- JDBC
- PostgreSQL
- MySQL

The actual implementation of a matrix becomes secondary!
Integration of Other Matrix Libraries

Switching to faster libraries for better performance. Example: SVD

using JAMA

switching to ojAlgo

relative performance

matrix size

JAMA
Colt
EJML
MTJ
ojAlgo
UJMP
There are three different „modes“ to perform a calculation.

- **original:**
  - Matrix
  - Calculation

- **copy:**
  - Matrix
  - Calculation
  - Matrix

- **link:**
  - Matrix
  - Calculation
## Matrix Annotation

Data requires annotation to be valuable.

<table>
<thead>
<tr>
<th>Report June 2009</th>
<th>product data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>product id</td>
</tr>
<tr>
<td>row1</td>
<td>6757</td>
</tr>
<tr>
<td>row2</td>
<td>6876</td>
</tr>
<tr>
<td>row3</td>
<td>9976</td>
</tr>
<tr>
<td>row4</td>
<td>9975</td>
</tr>
<tr>
<td>row5</td>
<td>980</td>
</tr>
<tr>
<td>row6</td>
<td>8657</td>
</tr>
<tr>
<td>row7</td>
<td>7677</td>
</tr>
<tr>
<td>row8</td>
<td>7657</td>
</tr>
<tr>
<td>row9</td>
<td>6678</td>
</tr>
<tr>
<td>row10</td>
<td>8865</td>
</tr>
</tbody>
</table>

**Label for row axis**

- row1
- row2
- row3
- row4
- row5
- row6
- row7
- row8
- row9
- row10
Automatic Entry Type Conversion

Not all matrices contain numerical data.

Matrix imported from CSV

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;This&quot;</td>
<td>&quot;5.7&quot;</td>
</tr>
<tr>
<td>&quot;matrix&quot;</td>
<td>&quot;1.9&quot;</td>
</tr>
<tr>
<td>&quot;contains&quot;</td>
<td>&quot;4.0&quot;</td>
</tr>
<tr>
<td>&quot;Strings, &quot;</td>
<td>&quot;1.2&quot;</td>
</tr>
<tr>
<td>&quot;data&quot;</td>
<td>&quot;9.1&quot;</td>
</tr>
<tr>
<td>&quot;must&quot;</td>
<td>&quot;0.5&quot;</td>
</tr>
<tr>
<td>&quot;be&quot;</td>
<td>&quot;7.7&quot;</td>
</tr>
<tr>
<td>&quot;converted&quot;</td>
<td>&quot;3.8&quot;</td>
</tr>
</tbody>
</table>

getAsString(3,1) → "1.2"
getAsDouble(3,1) → 1.2
getAsInt(3,1) → 1
getAsLong(3,1) → 11
getAsBoolean(3,1) → true

Supported value types: float, double, byte, char, short, int, long, boolean, Date, BigDecimal, BigInteger, String, Object, <Generic>
It is important to visualize data.

Example Code:

```java
Matrix m1, m2, m3, m4, m5, lu[], m6;
FileFormat csv = FileFormat.CSV;
File file = new File("example.csv");
m1 = MatrixFactory.importFromFile(csv, file);
m2 = m1.select(Ret.NEW, "4-100;6,7,8-10,100-200");
m3 = m2.standardize(Ret.NEW, Matrix.ROW);
m4 = m3.addMissing(Ret.NEW, Matrix.ALL, 0.05);
m5 = m4.impute(Ret.NEW, ImputationMethod.KNN, 2);
lu = m5.lu();
m6 = lu[0].bootstrap(Ret.LINK, 1000000);
double y = m6.getAsDouble(0, 0);
```
Summary and Discussion

The Universal Java Matrix Package: A novel and innovative matrix library for Java.

Summary

- Extendable architecture
- Ready for large amounts of data
- Integration of other libraries
- Flexible calculation methods
- Open Source LGPL
- Online forum for Q&A

Future Work:

- Documentation
- Developers wanted!

Homepage: http://www.ujmp.org