# Local and Distributed Matrices

# Lesson Objectives

- After completing this lesson, you should be able to:
  - Understand local and distributed matrices
  - Create dense and sparse matrices
  - Create different types of distributed matrices

# Local Matrices

- Natural extension of Vectors
- Row and column indices are 0-based integers and values are doubles
- Local matrices are stored on a single machine
- MLlib's matrices can be either dense or sparse
- Matrices are filled in column major order

# Dense Matrices

- A "reshaped" dense Vector
- First two arguments specify dimensions of the matrix
- Entries are stored in a single double array

# A Simple Dense Matrix

```
from pyspark.mllib.linalg import Matrix, Matrices
```

```
Matrices.dense(3, 2, [1, 3, 5, 2, 4, 6])
```

```
DenseMatrix(3, 2, [1.0, 3.0, 5.0, 2.0, 4.0, 6.0], False)
```

# Sparse Matrices in Spark: Compressed Sparse Column (CSC) format

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | | | | |
| | | 34 | | |
| | | | | |
| | | | 55 | |
| | | | | |

Rows: 5

Columns: 4

Column pointers: (0, 0, 1, 2, 2)

Row indices: (1, 3)

Non-zero values: (34, 55)

# Sparse Matrix Example

```
Matrices.sparse(5, 4, [0,0,1,2,2], [1,2], [34,55])
```

```
SparseMatrix(5, 4, [0, 0, 1, 2, 2], [1, 2], [34.0, 55.0], False)
```

# Distributed Matrices

- Distributed matrices are where Spark starts to deliver significant value
- They are stored in one or more RDDs
- Three types have been implemented:
  - RowMatrix
  - IndexedRowMatrix
  - CoordinateMatrix
- Conversions may require an expensive global shuffle

# `RowMatrix`

- The most basic type of distributed matrix
- It has no meaningful row indices, being only a collection of feature vectors
- Backed by an RDD of its rows, where each row is a local vector

# RowMatrix

- Assumes the number of columns is small enough to be stored in a local vector
- Can be easily created from an instance of RDD[Vector]

# A Simple `RowMatrix`

```
from pyspark.rdd import RDD
from pyspark.mllib.linalg.distributed import RowMatrix
```

```
rows = sc.parallelize([Vectors.dense(1.0,2.0),
                       Vectors.dense(4.0,5.0),
                       Vectors.dense(7.0,8.0)])
```

# A Simple `RowMatrix`

```
mat = RowMatrix(rows)
```

```
mat.numRows()
```

```
3L
```

```
mat.numCols()
```

```
2L
```

# IndexedRowMatrix

- Similar to a `RowMatrix`
- But it has meaningful row indices, which can be used for identifying rows and executing joins
- Backed by an RDD of indexed rows, where each row is a tuple containing an index (long-typed) and a local vector
- Easily created from an instance of `RDD[IndexedRow]`
- Can be converted to a `RowMatrix` by calling `toRowMatrix()`

# A Simple `IndexedRowMatrix`

```
from pyspark.mllib.linalg import Vector, Vectors
from pyspark.mllib.linalg.distributed import IndexedRow, IndexedRowMatrix
```

```
idx_rows = sc.parallelize([IndexedRow(0,Vectors.dense(1.0,2.0)),
                           IndexedRow(1,Vectors.dense(4.0,5.0)),
                           IndexedRow(2,Vectors.dense(7.0,8.0))])
```

```
idx_mat = IndexedRowMatrix(idx_rows)
```

```
idx_mat.rows.collect()
```

```
[IndexedRow(0, [1.0,2.0]), IndexedRow(1, [4.0,5.0]), IndexedRow(2, [7.0,8.0])]
```

# `CoordinateMatrix`

- Should be used only when both dimensions are huge and the matrix is very sparse
- Backed by an RDD of matrix entries, where each entry is a tuple `(i: Long, j: Long, value: Double)` where:
  - i is the row index
  - j is the column index
  - value is the entry value

# `CoordinateMatrix`

- Can be easily created from an instance of RDD[MatrixEntry]
- Can be converted to an `IndexedRowMatrix` with sparse rows by calling `toIndexedRowMatrix()`

# A Simple `CoordinateMatrix`

```
from pyspark.mllib.linalg.distributed import MatrixEntry, CoordinateMatrix
```

```
entries = sc.parallelize([MatrixEntry(0,0,9.0),
                          MatrixEntry(1,1,8.0),
                          MatrixEntry(2,1,6.0)])
```

```
coord_mat = CoordinateMatrix(entries)
```

```
coord_mat.toIndexedRowMatrix().rows.collect()
```

```
[IndexedRow(0, (2,[0],[9.0])),
 IndexedRow(1, (2,[1],[8.0])),
 IndexedRow(2, (2,[1],[6.0]))]
```

# Lesson Summary

- Having completed this lesson, you should be able to:
  - Understand local and distributed matrices
  - Create dense and sparse matrices
  - Create different types of distributed matrices