Introduction to Spark Machine Learning

Kenny Lu
School of Information Technology
Nanyang Polytechnic

October 9, 2017

What is Spark Machine Learning?

- A library of tools developed for machine learning
- Built over the Apache Spark framework

Why Spark Machine Learning?

- Scalable
- Compatibility (Java, R, Python, Scala)
- Data integration (HDFS, Cassendra, ...)

Spark Machine Learnng Data Types

- RDD
- Vector
- Labeled Points
- Matrix

Resilient Distributed Dataset

- RDD is an abstraction over a collection of data set being distributed and partitioned across a cluster of worker machines, mostly in memory.
- Programmers are not required to manage or to coordinate that distributed and partitioned. RDD is fault tolerant.
- RDDs are initialized and managed by the SparkContext.

Resilient Distributed Dataset

scala

```
val sc = new SparkContext("local", "shell")
// an RDD of doubles
val seriesX:RDD[Double] = sc
  .textFile("data/basic/series1.txt")
  .map(_.toDouble)
```

RDD transformations are pure

Recall from the previous section. Let `r` denotes an RDD,

- `r.map(f)` and `r.flatMap(f)` applies `f` to elements in `r`.
- `r.filter(f)` filters away elements `x` in `r` which `f(x)` yields `false`.
- assuming `r` is a collection of key-value pairs, `r.reduceByKey(f)` will shuffle the pairs and group them by keys. The values grouped under the same key will be reduced by `f`. Data locality is exploit when possible.

Localizing an RDD

An example of RDD

```
val sc = new SparkContext("local", "shell")
// an RDD of doubles
val seriesX:RDD[Double] = sc
  .textFile("data/basic/series1.txt")
  .map(_.toDouble)
val arr:Array[Double] = seriesX.collect.toArray
```

Vector

Vectors are local data collection.

- Dense vector - similar to an array. All values need to be specified.

Dense Vector

```
// Create a dense vector (1.0, 0.0, 3.0).
val dv: Vector = Vectors.dense(1.0, 0.0, 3.0)
```

Vector

Vectors are local data collection.

- Sparse vector - programmers are required to specify the size of the vector as well as the non-zero values.

Sparse Vector

```
// Create a sparse vector (1.0, 0.0, 3.0)
// by specifying its indices and values
// corresponding to nonzero entries.
val sv1: Vector = Vectors
  .sparse(3, Array(0, 2), Array(1.0, 3.0))
// Create a sparse vector (1.0, 0.0, 3.0)
// by specifying its nonzero entries.
val sv2: Vector = Vectors
  .sparse(3, Seq((0, 1.0), (2, 3.0)))
```

Labeled Point

Labeled points are vectors that with an assigned/labeled values. They are commonly used as the training data in algorithms such as logistic regressions and SVM. Labeled point is a labeled value (normally 0 or 1) with a vector.

Labeled points

```
// Create a labeled point with a positive label
// and a dense feature vector.
val pos = LabeledPoint(1.0, Vectors.dense(1.0, 0.0, 3.0))
// Create a labeled point with a negative label
// and a sparse feature vector.
val neg = LabeledPoint(0.0,
  Vectors.sparse(3, Array(0, 2), Array(1.0, 3.0)))
```

Matrix

Matrices can be seen as two diemsnional vectors.

- Local Matrix
- Distributed Matrix

Local Matrix

Local matrices are similar to vector

Local Matrix

```
// a 3 (rows) by 2 (columns) matrix
val dm: Matrix = Matrices
  .dense(3, 2, Array(1.0, 3.0, 5.0, 2.0, 4.0, 6.0))
```

Distributed Matrix

Distributed matrices are practically RDD of local vectors.

- RowMatrix
- IndexedRowMatrix
- CoordinateMatrix

RowMatrix

Row-oriented distributed matrix

RowMatrix

```
val rows: RDD[Vector] = MLUtils.loadVectors(sc,
  "data/basic/vector1.txt")
// Create a RowMatrix from an RDD[Vector].
val mat: RowMatrix = new RowMatrix(rows)
// Get its size.
val m = mat.numRows()
val n = mat.numCols()
```

IndexedRowMatrix

IndexedRowMatrix is an RowMatrix with indices (keys) to the rows.

IndexedRowMatrix

```
val indexedRows : RDD[IndexedRow] = rows
  .zipWithUniqueId().map( idv => IndexedRow(idv._2,idv._1)
// Create an IndexedRowMatrix from an RDD[IndexedRow].
val irmat: IndexedRowMatrix =
  new IndexedRowMatrix(indexedRows)
```

CoordinateMatrix

A CoordinateMatrix is a distributed matrix backed by an RDD of its entries. Each entry is a tuple of (i: Long, j: Long, value: Double), where i is the row index, j is the column index, and value is the entry value. A CoordinateMatrix is a sparse matrix

CoordinateMatrix

```
val local_entries : List[MatrixEntry] =
  (for ( i <- (1 to 10); j <- (2 to 5) )
  yield MatrixEntry(i, j, 0.0)).toList
val entries: RDD[MatrixEntry] =
  sc.parallelize(local_entries, 2)
// Create a CoordinateMatrix from
 // an RDD[MatrixEntry].
val cmat: CoordinateMatrix = new CoordinateMatrix(entries)
```

Machine learning Models

With above data types, we can build and use many interesting models. Spark ML comes with many builtin models,

- Classification
  - Logistic Regression
  - SVM
  - NaiveBayse
- Clustering
  - KMeans
  - LDA
- Regression
  - Linear Regression
- Other algorithms
  - PCA
  - NGram
  - Word2Vec

We will see some of these in the exercise

References

- The examples are adapted from
  https://spark.apache.org/docs/latest/
- More models of the library can be found
  https://spark.apache.org/docs/latest/api/scala/
  https://spark.apache.org/docs/latest/api/java/