Sampling

Lesson Objectives

- After completing this lesson, you should be able to:
  - Perform standard sampling on any RDD
  - Split any RDD randomly into subsets
  - Perform stratified sampling on RDDs of key-value pairs

Sampling

- Can be performed on any RDD
- Returns a sampled subset of an RDD
- Sampling with or without replacement
- Fraction:
  - –without replacement - expected size of the sample as fraction of RDD's size
  - –with replacement - expected number of times each element is chosen
- Can be used on bootstrapping procedures

A Simple Sampling

```
elements = sc.parallelize([Vectors.dense(4.0,7.0,13.0),
                           Vectors.dense(-2.0,8.0,4.0),
                           Vectors.dense(3.0,-11.0,19.0)])
```

```
elements.sample(withReplacement=False, fraction=0.5, seed=10L).collect()
```

```
[DenseVector([4.0, 7.0, 13.0]), DenseVector([-2.0, 8.0, 4.0])]
```

```
elements.sample(withReplacement=True, fraction=3.0, seed=10L).collect()
```

```
[DenseVector([4.0, 7.0, 13.0]),
 DenseVector([4.0, 7.0, 13.0]),
 DenseVector([-2.0, 8.0, 4.0]),
 DenseVector([-2.0, 8.0, 4.0]),
 DenseVector([-2.0, 8.0, 4.0]),
 DenseVector([3.0, -11.0, 19.0]),
 DenseVector([3.0, -11.0, 19.0])]
```

Random Split

- Can be performed on any RDD
- Returns an array of RDDs
- Weights for the split will be normalized if they do not add up to 1
- Useful for splitting a data set into training, test and validation sets

A Simple Random Split

```
data = sc.parallelize(range(1,1000000))
```

```
splits = data.randomSplit([0.6, 0.2, 0.2], seed = 13L)
training = splits[0]
test = splits[1]
validation = splits[2]
```

```
map(lambda rdd: rdd.count(), splits)
```

```
[601691, 198898, 199410]
```

Stratified Sampling

- Can be performed on RDDs of key-value pairs
- Think of keys as labels and values as an specific attribute

Stratified Sampling

- Two supported methods defined in PairRDDFunctions:
  - –sampleByKey requires only one pass over the data and provides an expected sample size
  - –sampleByKeyExact provides the exact sampling size with 99.99% confidence but requires significantly more resources

An Approximate Stratified Sampling

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

```
fractions = {0: 1.0, 1: 0.5}
```

```
approxSample = rows.map(lambda row: (row.index, row.vector)).sampleByKey(False, fractions, 2)
```

```
approxSample.collect()
```

```
[(0L, DenseVector([1.0, 2.0])), (1L, DenseVector([4.0, 5.0]))]
```

Lesson Summary

- Having completed this lesson, you should be able to:
  - Perform standard sampling on any RDD
  - Split any RDD randomly into subsets
  - Perform stratified sampling on RDDs of key-value pairs