

# Mining massive Datasets WS 2017/18

## Problem Set 1

Rudolf Chrispens, Marvin, Daniela Schacherer

October 30, 2017

### Exercise 01

Given is a cluster of  $n$  machines, each having a probability  $p$  of failing.

- a) The probability of one machine to not fail is  $1 - p$ .

The probability of ALL machines not failing is  $n$  times  $1 - p$  which is  $(1 - p)^n$ .

The probability of at least one machine failing is the opposite event and thus  $1 - (1 - p)^n$ .

- b) The probability  $p_k$  of exactly  $k$  machines failing can be described using the binomial distribution. The binomial distribution describes the discrete probabilities of the number of successes in a sequence of independent experiments. As we have independent machines in the cluster with the number of successes corresponding to a machine failing we can write:

$$p(k|p,n) = \binom{n}{k} p^k (1 - p)^{n-k}$$

$p^k$  is the probability that  $k$  machines fail which has to be multiplied to the probability that the other  $n - k$  machines do not fail. The binomial coefficient is the combinatoric element and describes in which way  $k$  elements can be chosen from  $n$  elements.

- c) Zz.:  $p_1 + p_2 + \dots + p_n = 1 - (1 - p)^n$

We have  $p_1 = p_2 = \dots = p_n = p = \binom{n}{k} p^k (1 - p)^{n-k}$

$$p_1 + p_2 + \dots + p_n = \sum_{k=1}^n \binom{n}{k} p^k (1 - p)^{n-k}$$

We can use the binomial theorem:  $\sum_{k=0}^n \binom{n}{k} y^k x^{n-k} = (x + y)^n$  but have to subtract  $p_0$  again

$$\begin{aligned}
 &= \sum_{k=0}^n \binom{n}{k} p^k (1-p)^{n-k} - \binom{n}{0} p^0 (1-p)^n \\
 &= ((1-p) + p)^n - (1-p)^n \\
 &= 1^n - (1-p)^n \\
 &= 1 - (1-p)^n
 \end{aligned}$$

## Exercise 02

a1) -join() - TRANSFORMATION

**Input:** otherDataset, [numTasks]

**Output:** Returns a dataset with "Key/(V1,V2)" pairs.

**Code Example** `rdd1 = sc.parallelize([("foo", 1), ("bar", 2), ("baz", 3)])`  
`rdd2 = sc.parallelize([("foo", 4), ("bar", 5), ("bar", 6)])`  
`rdd1.join(rdd2)`

a2) -sort() - TRANSFORMATION - Could not find sort() in reference used sortByKey() instead  
[-https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html](https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html)

**Input:** [ascending], [numTasks]

**Output:** When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.

**Code Example:** `names = sc.textFile(sys.argv[1])`  
`filtered_rows = names.filter(lambdaline : "Count" not inline).map(lambdaline : line.split(","))`  
`filtered_rows.map(lambdan : (str(n[1]),int(n[4]))).sortByKey().collect()`

a2) -groupby() - TRANSFORMATION - Could not find groupby() in reference used groupByKey()  
[instead -https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html](https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html)

**Input:** [ascending], [numTasks]

**Output:** Returns a dataset with "Key/Value" Pairs sorted ascending or descending.

**Code Example:**

```
lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
words = lines.flatMap(lambda x: x.split(' '))
words.reduceByKey(lambda x, y: x + y, 5)
words.groupByKey(5)
```

b1) -NOTE All the tested source code is in *U1Ex2.py*

-INTERSECTION

**Input:** [RDD]

**Output:** Returns a RDD with the intersecting elements of two datasets.

**Code example:**

```
intersectRDD1 = sc.parallelize(range(1, 10))
intersectRDD2 = sc.parallelize(range(5, 15))
intersect = intersectRDD1.intersection(intersectRDD2).collect()
print(intersect)
```

**exampleOutput:** [8, 9, 5, 6, 7]

b2) -DISTINCT

**Input:** [numTasks]

**Output:** Return a new dataset that contains the distinct elements of the source dataset.

**example Code:**

```
distinctRDD1 = sc.parallelize(range(1, 12))
distinctRDD2 = sc.parallelize(range(8, 20))
distinct = distinctRDD1.union(distinctRDD2).distinct().collect()
print(distinct)
```

**exampleOutput:** [8, 16, 1, 9, 17, 2, 10, 18, 3, 11, 19, 4, 12, 5, 13, 6, 14, 7, 15]

b3) -UNION

**Input:** [RDD]

**Output:** Return a new dataset that contains the union of the elements in the source dataset and the argument.

**example Code:** `unionRDD1 = sc.parallelize(range(1, 7))  
unionRDD2 = sc.parallelize(range(3, 10))  
union = unionRDD1.union(unionRDD2).collect()  
print(union)`

**exampleOutput:** `[1, 2, 3, 4, 5, 6, 3, 4, 5, 6, 7, 8, 9]`

b4) -COLLECT

**Input:** NONE is called as a function on an RDD

**Output:** Return all the elements of the dataset as an array at the driver program.

**example Code:** `collection = sc.parallelize([1, 2, 3, 4, 5]).flatMap(lambda x: [x, x, x]).collect()  
print(collection)`

**exampleOutput:** `[1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5]`

b5) -COUNT

**Input:** NONE is called as a function on an RDD

**Output:** Return all the number of elements of the dataset as an array at the driver program.

**example Code:** `names1RDD = sc.parallelize(["Daniela", "Marvin", "Rudolf", "Kevin", "Jaqueline"])  
counts = names1RDD.count()  
print(counts)`

**exampleOutput:** `5`

b6) -FIRST

**Input:** NONE is called as a function on an RDD

**Output:** Return all the first element of the dataset as an array at the driver program.

**example Code:** `names2RDD = sc.parallelize(["Daniela", "Marvin", "Rudolf"])  
first = names2RDD.first()  
print(first)`

**exampleOutput:** Daniela

## Exercise 03

a) see comments in `01-03kmeans.py` see `01 – 03kmeans.py`

## Exercise 05

b) Version 1: `[:]` is missing in line 130

In line 130 the variable `centroids` and `newCentroids` would refer to the same instance. In the for-loop `newCentroids` is changed and a new instance with the same values is created with `centroids = newCentroids[:]` in line 157.

- Version 2: `[:]` is missing in line 157

In line 130 `centroids` and `newCentroids` will refer to different instances. In the for-loop `newCentroids` is changed and the variable `centroids` is in line 157 assigned to `newCentroids`, meaning they then refer to the same object. In every further for-loop `newCentroids` will be changed and then assigned to `centroids` although they are already the same instance.

- Version 3: `[:]` is missing in line 130 and line 157

In line 130 the variable `centroids` and `newCentroids` would refer to the same instance. In the for-loop `newCentroids` is changed which changes also `centroids` as they refer to the same object. The same is true for every further for-loop. One of the two variables is therefore needless.