Mining Massive Datasets
Solutions to Problem Set 1

Due: 28.10.2019, 11:59 pm

Duc Anh, Phi 3550091

Mustafa, Ibrahim 3284705

Amrit, Sandhar

__Exercise 1__

Consider a cluster of ***n*** machines. Each has a probability ***p*** of failing in a given period of time ***T*.**

a) What is the probability of at least one machine failure during this period of time?

Solution:
Use Binomial Distribution:

p = probability of failure during time period T
n = number of machines
k = number of machines failing

$$p_k = p^k * (1-p)^{n-k} * \left(\frac{n!}{k! * (n-k)!}\right)$$
$$p_1 = p * (1-p)^{n-1} * n$$

b) For $0 \leq k \leq n$, what is the probability p_k (in terms of *k, n* and *p*), of exactly *k* machines failing during T? Give an explanation.

Solution:
Use Binomial Distribution:

$$p_0 = 1 * (1-p)^n * 1$$
$$p_k = p^k * (1-p)^{n-k} * \left(\frac{n!}{k! * (n-k)!}\right)$$
$$p_n = p^n * 1 * 1$$

$$(1-p)^n \leq p_k \leq p^n$$

The binomial distribution formula consists of:

- p^k → This term shows the probability of *k* machines failing during *T*
- $(1-p)^{n-k}$ → This term shows the complementary probability of *n-k* machines not failing during *T*
- $\left(\frac{n!}{k! * (n-k)!}\right)$ →This term shows the possible combinations *k* machines can fail out of *n* machines
- altogether these terms calculate the probability p_k of exactly k machines failing during T

c) Show that the probabilities from **b)** satisfy:
$$p_1 + p_2 + ... + p_n = 1 - (1-p)^n$$

Solution:

$$p_0 + p_1 + p_2 + ... + p_n = 1$$
$$p_0 = (1-p)^n$$

<=> $p_1 + p_2 + ... + p_n = 1 - (1-p)^n$

<u>Exercise 2</u>

a)

join(*otherDataset*, [*numTasks*]):

When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key.

Example:
input:

```
    firstNames = [(0, Alex), (1, Anton), (2, Bob), (3, Chandler)]
    lastNames = [(0, Armin),(1, Brech),(1, Krug),(2, Kraft),(3, Bing)]
```

output:

```
    firstNames.join(lastNames):
        [
            (0, (Alex, Armin)),
            (1, (Anton, Brech)),
            (1, (Anton, Krug),
            (2, (Bob, Kraft)),
            (3, (Chandler, Bing)
        ]
```

sort():
There is no sort() function for RDD's (only for Dataframes), however the **sortby(**keyfunc, ascending=True, numPartitions=None**)** function sorts the RDD using a key function.

Example:
input:

```
    letters = [('c', 1),('a', 3),('d', 6),('e', 15),('b', 10)]
```

output:

```
    letters.sortby(lambda letter: letter[0]):
        [(‘a’, 3),(‘b’, 10),(‘c’, 1),(‘d’, 6),(‘e’, 15)]
    letters.sortby(lambda letter: letter[1], ascending=False):
        [(‘e’, 15),(‘b’, 10),(‘d’, 6),(‘a’, 3),(‘c’, 1)]
```

groupBy(f, numPartitions=None, partitionFunc=<function portable_hash>)
Creates keys for each entry using function f, then groups all entries with the same key as the value to that key

Example:
input:
```
    numbers = [1, 2, 3, 4, 5, 6, 7, 8]
```
output:
```
    numbers.groupBy(lambda num: num % 2):
        [(0, [2, 4, 6, 8]), (1, [1, 3, 5, 7])]
```

Exercise 3

a)
What does broadcast provide?

Broadcast provide a way to share a readonly variable across multiple machines for task and operations efficiently. It sends the variable to the machines only once where it is cached respectively.

Which other mechanism does it improve and how?

It can reduce number of send data over the network and thus keep communication cost low. An example is shown for the join operation for two datasets. Normally you would directly join both datasets, which shuffles both over the network. A better approach would be, if one dataset is particularly small, to broadcast the smaller dataset as a map to all machines which contain the other dataset, in order to perform the join operation. This way, only the smaller dataset is send over the network.

Which features of the distributed program determine the number of times the variable will be actually transmitted over the network? Explain the role of tasks and nodes in this.

By default, when Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. The number of times the variable will be actually transmitted over the network depends on the number of tasks which depend on the number of partitions. However, if a variable is broadcasted, it is only send once for each node in the cluster and is going to be cached respectively.

b)

Describe the function of an accumulator.

Often, an application needs to aggregate multiple values as it progresses.
Accumulators provide a simple syntax for aggregating values from worker nodes back to the driver program.

What is the alternative implementation without an accumulator and why is an accumulator a preferred option?

You could use the reduce() function alternatively.
The problem with reduce() is that you only get the result after all data has been processed. In contrast the accumulator mechanism gets real-time updates on its accumulator variable from each worker node. It can also aggregate multiple variables across multiple tasks.

Which example application for an accumulator is discussed in the video?

Overall count of bad records and bad bytes after filtering records. Additionally it can be used for computing averages.

What has to be implemented in order to define a custom accumulator?

Define an object extending AccumulatorParam\[T\], where T is your data type, and tell the system how to work with a custom data type T. Define default value to be initialized for a given T. And define an addInPlace method to merge in values.

Compare the accumulator mechanism to the reduce()-function

The reduce()-function collects data from all RDDs and reduces them to one value. The problem is that you only get the result after all data has been processed. In contrast the accumulator mechanism gets real-time updates on its accumulator variable from each worker node. Additionally the reduce function is limited to one variable which is accumulated, whereas with the accumulator mechanism you can define multiple variables which you could accumulate on in parallel, even across multiple tasks.

c)

Give three examples of RDD operators that result in RDDs with partitioning.

- join()
- mapValues()
- reduceByKey()

Explain the connection between partitioning and network traffic.

If data is spread across machines arbitrarily (partitions) and if the data has to come together on the same machine there is a lot of network traffic.

How does the modification on the pageRank example use partitioning to make the code more efficient?

Prepartition the links RDD so that links for URLs with the same hash code are on the same node. This saves future shuffling.

How does Spark exploit the knowledge about the partitioning to save time in task execution?

It arranges how data is spread across machines in a way, so that data with the same key is on the same machine. This reduces the amount of data which needs to be send over the network, which then saves time in task execution because the tasks receive the data quicker.

How can you create a custom Partitioner?

You can define your own subclass of Partitioner to leverage domain-specific knowledge. This subclass must contain three function:

- numPartitions: defines the number of partitions
- getPartition: get a partition by key
- equals: tell whether two partitions are equal