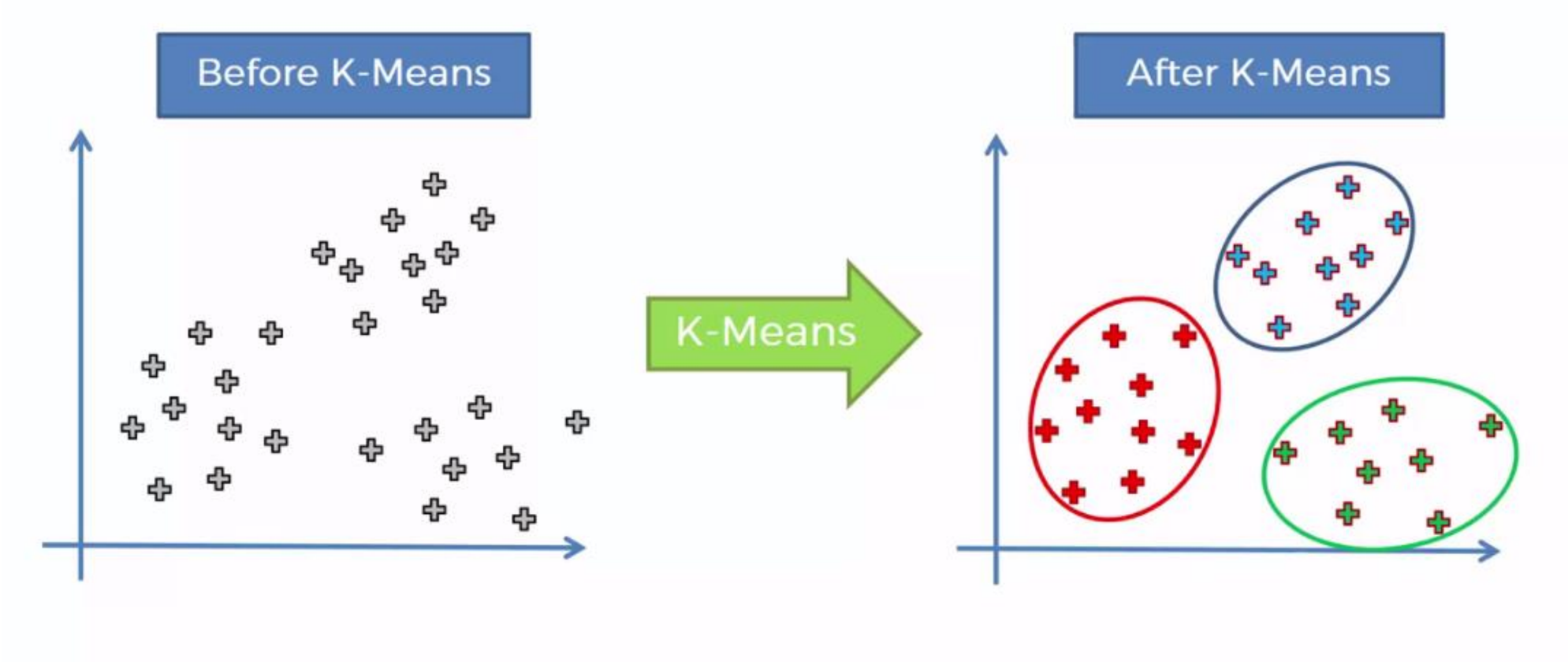
Practice 6

K-Means

Problem

- **Use K-means in mllib**
  - Use predefined function in pyspark.**mllib.clustering**



https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.clustering.KMeans

Dataset for K-means

➢ Recognition of handwritten digits

- There are 10 handwritten digits(0~9) in bitmap format.

➢ 64 Features (pixel values)

1. Pixel 1
2. Pixel 2
…
…
…
64. Pixel 64
65. digit

❖ The last column of the data matrix indicates the class labels.

*** UCI Machine Learning Repository :**

https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits

➢ You can download the pre-processed dataset on iCampus

Practice 6

1. Use predefined classes in *pyspark.mllib.clustering : Kmeans()*

   Parameters for the method

   - *k=10, maxIterations=100, seed = given index*

2. Perform k-means 30 times and find the k-means model with the smallest objective

   function value. For example, like following.

```
kmeans_list = []
for i in range(30):
    kmeans_list.append(KMeans.train(trData,k=10,maxIterations=100, seed=i))
```

3. Then, with that model, calculate ***NMI score*** of the result to the test data points.

https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.clustering.KMeans

https://spark.apache.org/docs/latest/mllib-clustering.html

Submission

1. You have to submit "**result.txt**" file on iCampus.
2. In your **result.txt** file, there must be ***NMI score*** of K-Means clustering result for digit dataset.
3. ***NMI*** means ***normalized mutual information*** which is a metric to measure some clustering results.
4. Deadline: May 21st 23:59 P.M.
5. Your result.txt file must be like following

```
NMI of K-Means clustering
0.7499
```

Windows

```
NMI of K-Means clustering
0.7499
```

Linux

Solution

- Import libraries

```
from pyspark import SparkConf, SparkContext
from pyspark.mllib.clustering import KMeans
from sklearn.metrics.cluster import normalized_mutual_info_score as NMI
import math
```

To calculate K-Means objective function value

- Define functions

Change the data to be space separated

```
def parseFeat(line):
    values = [float(x) for x in line.replace(',',' ').split(' ')]
    return values[:-1]

def parseLabel(line):
    values = [float(x) for x in line.replace(',',' ').split(' ')]
    return values[-1]
```

Solution

K-Means model: It has sub-function

1) centers: return the center of predicted class
2) predict: return the predicted class of data points

- Define error function

```
def error(point, model):
    center = model.centers[model.predict(point)]
    return math.sqrt(sum([x**2 for x in (point-center)]))
```

- Configure Spark context

Set configuration of Spark, master as local

```
conf = SparkConf()
conf.set("spark.master", "local")
sc = SparkContext(conf=conf)
```

Solution

➢ Load training and test data points

```
data = sc.textFile("practice6_train.csv")
trData = data.map(parseFeat)

data = sc.textFile("practice6_test.csv")
tsData = data.map(parseFeat)
tsLabel = data.map(parseLabel)
```

Load dataset on iCampus.
If you have socket error, during loading dataset, then you must increase numPartition parameter in textFile function

Apply parseFeat function on RDD data

➢ Save K-Means model using different seeds

```
kmeans_list = []
for i in range(30):
    kmeans_list.append(KMeans.train(trData,k=10,maxIterations=100, seed=i))
```

To save K-Means model

Solution

- Find K-means model which has minimum objective function value

```
obj_list = []
for i in range(30):
    obj_list.append(trData.map(
        lambda point: error(point, kmeans_list[i])).reduce(lambda x,y: x+y))
```

Apply error function and get objective function value for each K-Means model

Sum of objective function value of all data points

- Using that model, predict the clustering result of test data points

```
kmeans = kmeans_list[obj_list.index(min(obj_list))]
tsPredict = kmeans.predict(tsData)
```

Solution

- Calculate NMI score of the result, and Stop Spark context

```
nmi_score = NMI(list(tsPredict.collect()), list(tsLabel.collect()))

f = open('result.txt','w')
f.write('NMI of K-Means clustering\n')
f.write('{:.4f}'.format(nmi_score))

sc.stop()
```

Calculate NMI score of the prediction for test data points

If your program runs without problem, then you must stop SparkContext

Solution

➢ **Result**

- **Deadline: *May 21st 23:59*. We don't allow late submissions.**
- **Your result might be like the following**

```
NMI of K-Means clustering
0.7499
```

Windows

```
NMI of K-Means clustering
0.7499
```

Linux