MapReduce

lec 05

K-Mean

- K-mean is unsupervised machine learning algorithm which groups the unlabelled dataset into different clustures based on their similarity.
- k-means clustering divides data into a predefined number of clusters.
- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

K-mean Algorithm

- **Initialization:** Choose K initial centroids randomly from the dataset. These centroids are the initial cluster centers.
- **Assignment:** For each data point in the dataset, calculate the distance (usually Euclidean distance) between that point and each of the K centroids. Assign the data point to the cluster whose centroid is closest to it.
- **Update:** Recalculate the centroids of each cluster by taking the mean of all data points assigned to that cluster.

K-mean Algorithm

- **Repeat:** Repeat the assignment and update steps until a convergence criterion is met. Common convergence criteria include when the centroids no longer change significantly, or when a fixed number of iterations is reached.
- **Result:** The final centroids represent the centers of the K clusters, and each data point belongs to the cluster associated with the nearest centroid.

Example

- $dist = \surd(\mu - x)^2$

Dataset		centroid 1 (3)	centroid 2(11)	cluster
1	2	1	9	1
2	4	1	7	1
3	10	7	1	2
4	12	9	1	2

cluster 1={2,4}

cluster 2={10,12}

New centroid 1: (2+4)/2=3

new centroid 2: (10+12)/2= 11

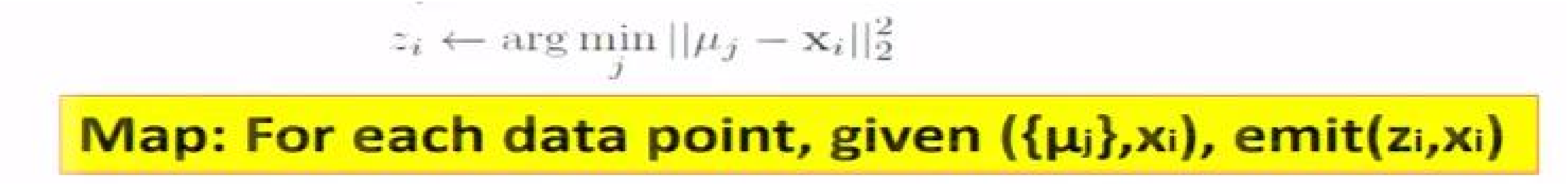
the centroid remain same so the results is

cluster 1={2,4,3} with centroid 3

cluster 2={10,12,11} with centroid 11

Parallel K-mean using MapReduce

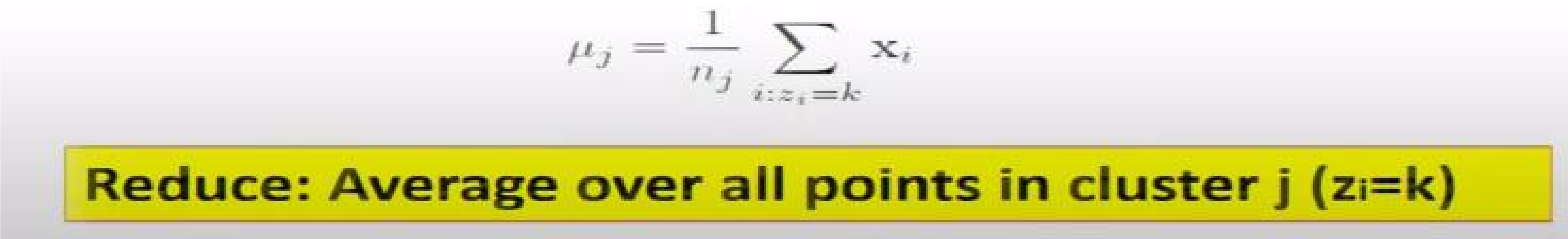
- **Mapper:**
  - assign data points to closest cluster center.



$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

Map: For each data point, given ({μ_j},x_i), emit(z_i,x_i)

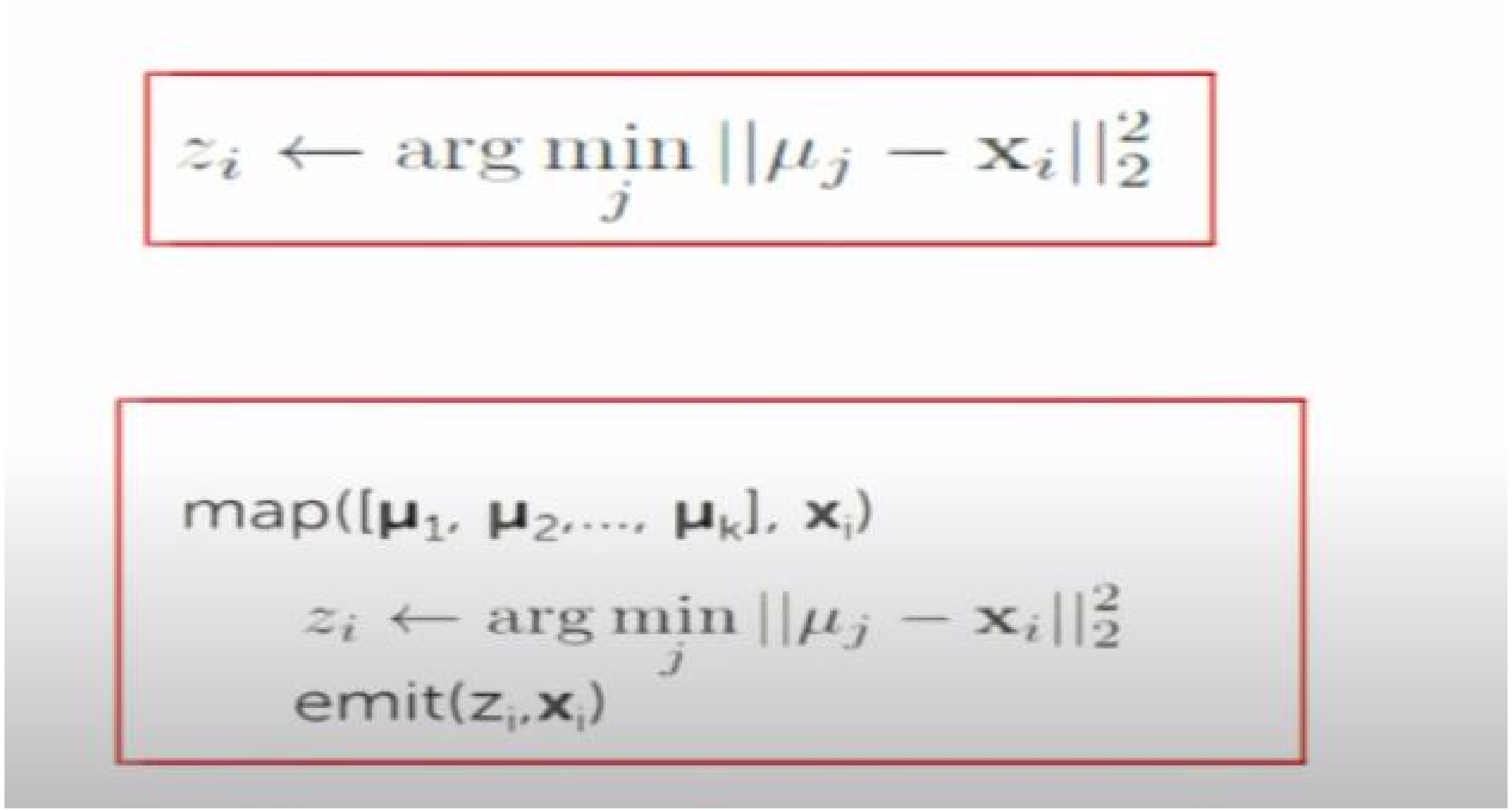
- **Reducer:**
  - revise cluster center as mean of assigned observation.



$$\mu_j = \frac{1}{n_j}\sum_{i:z_i=k} \mathbf{x}_i$$

Reduce: Average over all points in cluster j (z_i=k)

Mapper

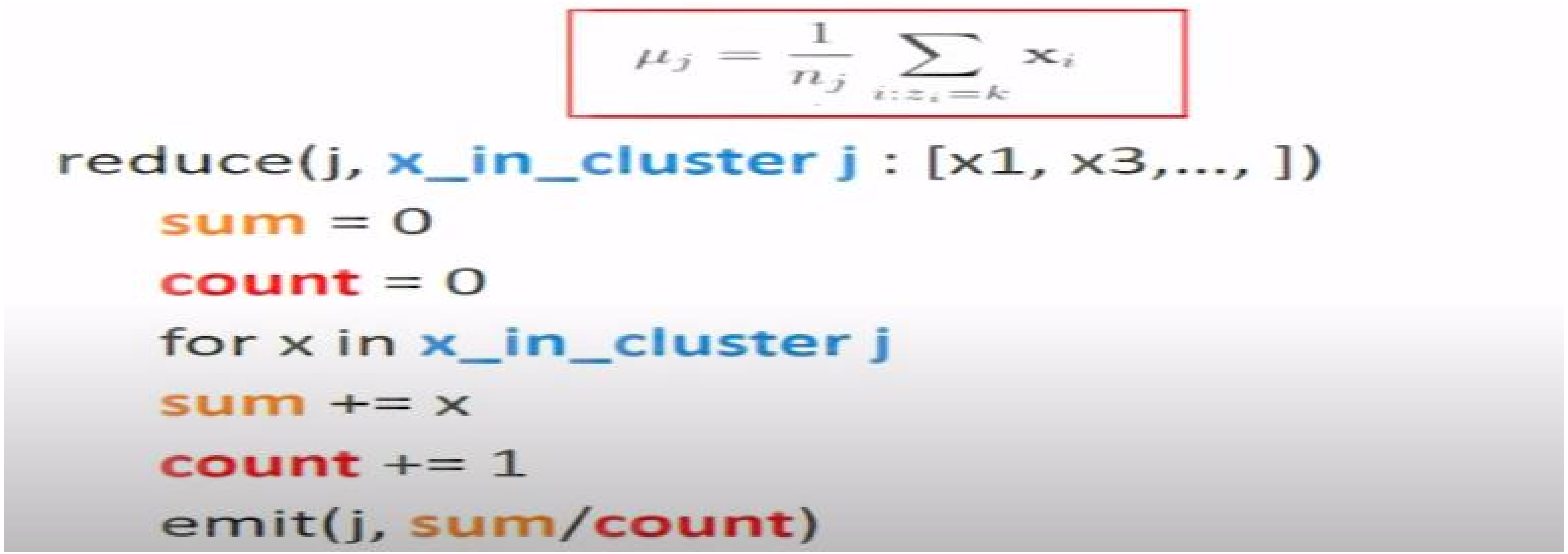


$$z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$$

```
map([$\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$,…, $\boldsymbol{\mu}_k$], $\mathbf{x}_i$)
    $z_i \leftarrow \arg\min_j ||\mu_j - \mathbf{x}_i||_2^2$
    emit($z_i$,$\mathbf{x}_i$)
```

Reducer

revise cluster center as mean of assigned observation



$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=k} \mathbf{x}_i$$

```
reduce(j, x_in_cluster j : [x1, x3,..., ])
    sum = 0
    count = 0
    for x in x_in_cluster j
    sum += x
    count += 1
    emit(j, sum/count)
```

K-mean using MapReduce

```
from mrjob.job import MRJob
import numpy as np

class ClusterAssignmentMapReduce(MRJob):

    def configure_args(self):
        super(ClusterAssignmentMapReduce, self).configure_args()
        self.add_file_arg('--clusters', help='File containing cluster centroids')
```

```
def mapper_init(self):
        # Load cluster centroids from the file
        with open(self.options.clusters, 'r') as f:
            self.cluster_centroids = [float(line.strip()) for line in f]

    def mapper(self, _, line):
        # Parse the input data point
        data_point = float(line.strip())

        # Find the nearest cluster for the data point
        nearest_cluster = min(self.cluster_centroids, key=lambda centroid: abs(centroid - data_point))

        # Emit the nearest cluster and the data point
        yield nearest_cluster, data_point
    def reducer(self, cluster, data_points):
        # Collect and emit data points for each cluster
        yield cluster, list(data_points)

if __name__ == '__main__':
    ClusterAssignmentMapReduce.run()
```