CLOUD COMPUTING APPLICATIONS

Spark MLlib

Roy Campbell & Reza Farivar

Spark MLlib

- Spark’s machine learning (ML) library
  - Ease of Use
  - Scalable

Collection of ML Libraries: Classification and Regression

- linear models (SVMs, logistic regression, linear regression)
- naive Bayes
- decision trees
- ensembles of trees (Random Forests and Gradient-Boosted Trees)

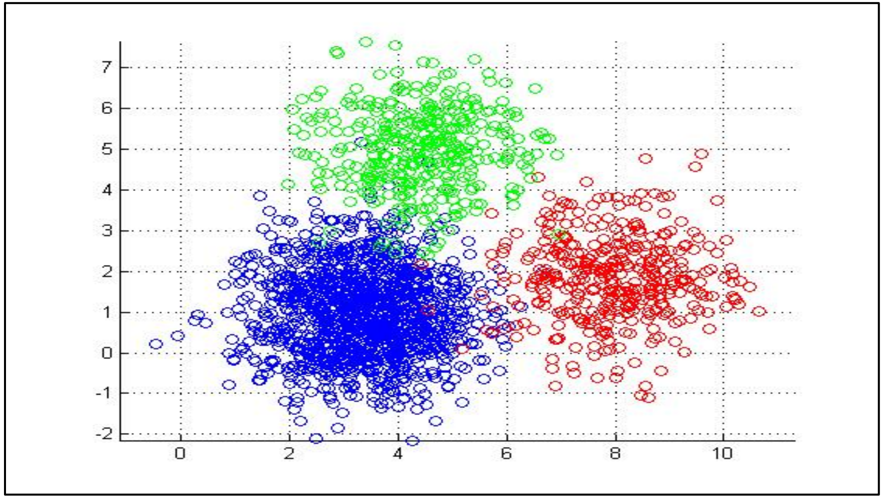
Collection of ML Libraries: Clustering

- k-means
- Gaussian mixture
- power iteration clustering (PIC)
- latent Dirichlet allocation (LDA)
- streaming k-means

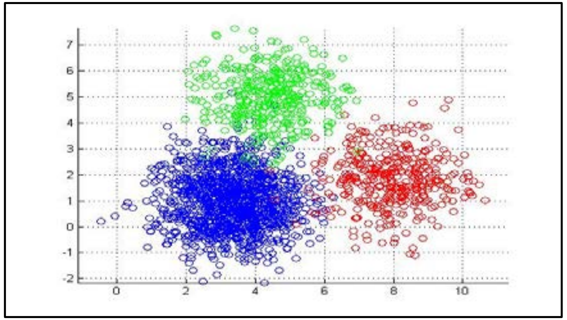
Collection of ML Libraries: Dimensionality Reduction

- Singular Value Decomposition (SVD)
- Principal Component Analysis (PCA)

Example: K-Means Clustering



K-Means in MapReduce

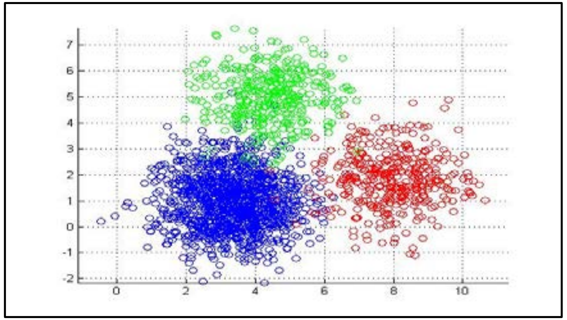


- **Input**
  - Dataset (set of points in 2D) --Large
  - Initial centroids (K points) --Small

- **Map Side**
  - Each map reads the K-centroids + one block from dataset
  - Assign each point to the closest centroid
  - Output <centroid, point>

K-Means in MapReduce (Cont’d)

- **Reduce Side**
  - Gets all points for a given centroid
  - Re-compute a new centroid for this cluster
  - Output: <new centroid>
- **Iteration Control**
  - Compare the old and new set of K-centroids
    - If similar ➔ Stop
    - Else
      - If max iterations has reached ➔ Stop
      - Else ➔ Start another Map-Reduce Iteration



K-Means Clustering in Spark

```
import org.apache.spark.mllib.clustering.{KMeans, KMeansModel}
import org.apache.spark.mllib.linalg.Vectors

// Load and parse the data
val data = sc.textFile("data/mllib/kmeans_data.txt")
val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble))).cache()

// Cluster the data into two classes using KMeans
val numClusters = 2
val numIterations = 20
val clusters = KMeans.train(parsedData, numClusters, numIterations)

// Evaluate clustering by computing Within Set Sum of Squared Errors
val WSSSE = clusters.computeCost(parsedData)
println("Within Set Sum of Squared Errors = " + WSSSE)

// Save and load model
clusters.save(sc, "myModelPath")
val sameModel = KMeansModel.load(sc, "myModelPath")
```