**Problem solving using linear regression.**

spark program for the following Machine Learning Tasks. A group of primatologists wants to study the details of the daily movement, activities, and interactions of a group of 6 chimpanzees living on "chimp island" - a natural, though somewhat open habitat about 50 meters in diameter, bounded on all sides by water, in the San-Diego zoo. Since they don’t want to sit all day every day recording the second-by second positions and activities of the chimps, they have come to you, a computer vision expert, for automated assistance. They are interested in both compiling statistics about the movement and location of individuals, and in the frequency and locations of different interactions and activities (feeding, sleeping, grooming, fighting, etc.) They are willing to help in labeling relevant activities, even to the point of answering a few hundred quick questions per day of data (what’s she doing here?), but they don’t want to sit through 24 hours of video to do it. Ultimately they want an automated database that they can use to find out how many hours a day chimp Jane sleeps and where, histogram preferred eating locations, obtain statistics on who grooms whom, etc.

**1.Implement to build a linear regression model for selected two parameters for chimpanzee’s daily movement, activities and interaction. Define your own datasets.**

We considered our variables as

1.Number of hours chimp sleeps. 2.Time it spent interacting with the visitors (hours). 3.Frequency of visit of a place in its cage. 4.Time in minutes it shouts.

predictor: How active the chip is! on the scale we defined

sample in data set:

(Each value is normalized with the mean of each its own set of values )

X- 1.2056 y- 0.2403,0.2966,0.5009,2.6503

**Data set division :**

randomSplit(Array(0.65, 0.35))

Parameters: Step size and iteration number are main parameter to be considered for moving out from local minimum and over fitting with the data set. Tried with many combinations of iterations and step size of loss function (MSE) . Data set is divided by a random spits , that’s difficult to find the optimal weights for model and Considered the minimum Mse after many iterations.

Linear regression code:

val sc=new SparkContext(sparkConf)

// Turn off Info Logger for Consolexxx

Logger.getLogger("org").setLevel(Level.OFF);

Logger.getLogger("akka").setLevel(Level.OFF);

// Load and parse the data

val data = sc.textFile("data\\lpsa.data")

val parsedData = data.map { line =>

val parts = line.split(',')

LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split(' ').map(\_.toDouble)))

}.cache()

parsedData.take(1).foreach(f=>println(f))

// Split data into training (95%) and test (5%).

val Array(training, test) = parsedData.randomSplit(Array(0.65, 0.35))

// Building the model

val numIterations = 100

val stepSize = 0.0001

val model = LinearRegressionWithSGD.train(training, numIterations, stepSize)

// Evaluate model on training examples and compute training error

val valuesAndPreds = training.map { point =>

val prediction = model.predict(point.features)

(point.label, prediction)

}

val MSE = valuesAndPreds.map{ case(v, p) => math.pow((v - p), 2) }.mean()

println("training Mean Squared Error = " + MSE)

// Evaluate model on test examples and compute training error

val valuesAndPreds2 = test.map { point =>

val prediction = model.predict(point.features)

(point.label, prediction)

}

val MSE2 = valuesAndPreds2.map{ case(v, p) => math.pow((v - p), 2) }.mean()

println("test Mean Squared Error = " + MSE2)

Final result:

training Mean Squared Error = 7.382755107707165 test Mean Squared Error = 7.569801455837684

**2.Implement K-Means clustering for the clusters of the chimpanzee’s activities. Define your own data sets.**

Recognizing the activity the chimp do, Classifying weather it as sleeping , eating and moving . Considering 3 different classes and applying k nearest neighbors to find the clusters which help in estimating its activity.

sample data in data set:

2 14 33 50

val numClusters = 3 val numIterations = 30

sample output:

([13.0,44.0,23.0,63.0],2) ([15.0,49.0,25.0,63.0],1) ([11.0,30.0,25.0,51.0],2) ([21.0,54.0,31.0,69.0],0)

k-mean clustering

val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[\*]")

val sc=new SparkContext(sparkConf)

// Turn off Info Logger for Consolexxx

Logger.getLogger("org").setLevel(Level.OFF);

Logger.getLogger("akka").setLevel(Level.OFF);

// Load and parse the data

val data = sc.textFile("data/test1.txt")

val parsedData = data.map(s => Vectors.dense(s.split(' ').map(\_.toDouble))).cache()

//Look at how training data is!

parsedData.foreach(f=>println(f))

// Cluster the data into two classes using KMeans

val numClusters = 3

val numIterations = 30

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)

//Look at how the clusters are in training data by making predictions

println("Clustering on training data: ")

clusters.predict(parsedData).zip(parsedData).foreach(f=>println(f.\_2,f.\_1))

**3)summarizing the annotations from clarifiai api**

workflow :

1) video data → frames. 2) key-feature(sift) → Most dissimilar frames. 3) Annotations to each frame - clarifi ai. 4) summarizing the output.

Summarizing output by words resulting top 6 words in each mainframe :

output :

**output\mainframes\354\_0.0025484199796126403.jpg\***\*

grass

nature

outdoors

no person

wildlife

wild