UNIVERSIDAD DE GRANADA

Escuela de Posgrado
Universidad de Granada

MLlib Noise and Instance Selection

Big Data II

Diego García (djgarcia@decsai.ugr.es) 19/04/2018

Outline

- **Catching up**
- Introduce Noise
- Noise Filters
- Instance Selection

Cache

RDD.cache() or RDD.persist()

Use it **BEFORE** an action

RDD.unpersist()

Upload files to cluster

To upload the file archivo.txt from our computer to /home/user folder, we do the following:

scp file.jar user@hadoop.ugr.es:/home/user

Download files from cluster

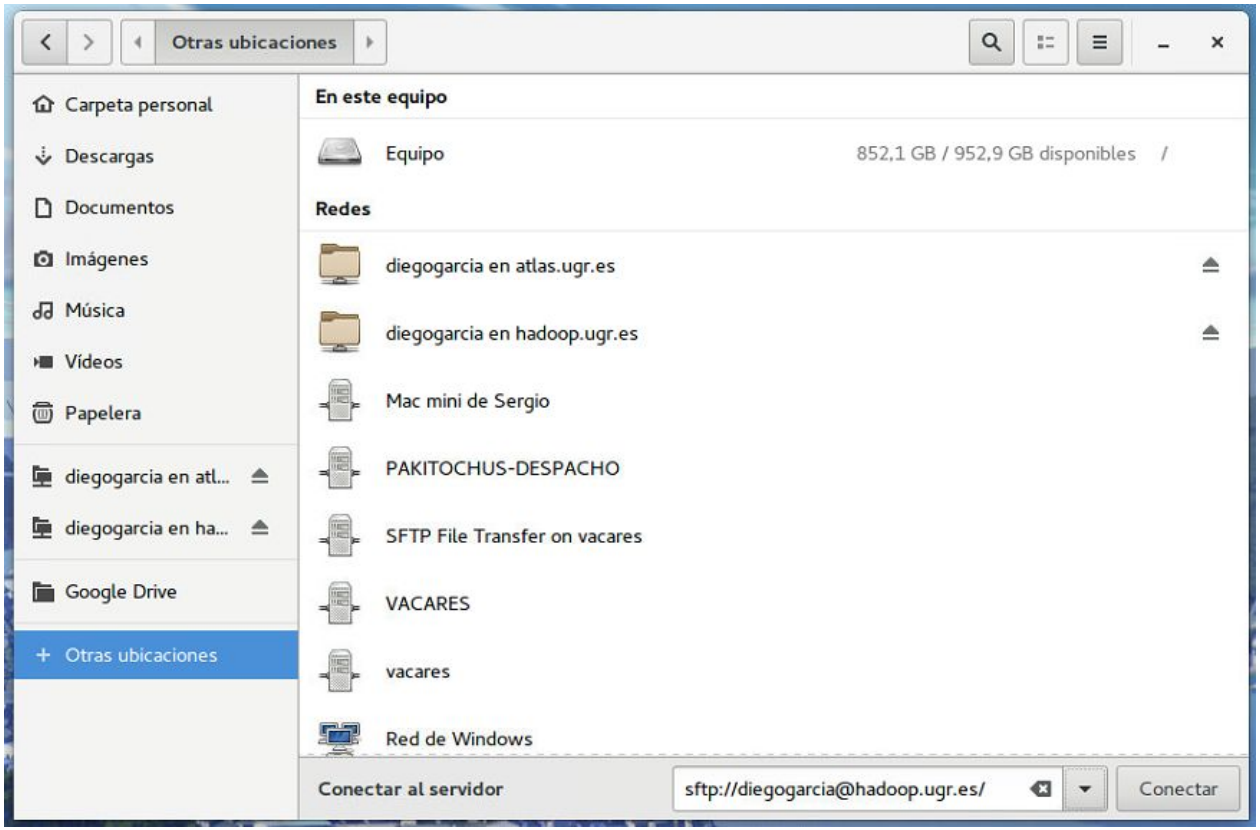
To download the file archivo.txt from the cluster to our computer in Docs folder, we do the following:

scp user@hadoop.ugr.es:/home/user/archivo.txt Docs

Alternatives

FileZilla

Nautilus



Run jobs in cluster

Always use **spark-submit** in cluster

/opt/spark-2.2.0/bin/spark-submit file.jar

Limit resources (workers/memory)

/opt/spark-2.2.0/bin/spark-submit --total-executor-cores 10 --executor-memory 10g file.jar

Compile Projects

Ensembles & Noise_IS

sh clean.sh

sh compile.sh #Creates target folder

sh run.sh #Local

Upload to Cluster

"Fat" .jar inside target folder

run.sh

#Cluster (.jar must be in cluster)

/opt/spark-2.2.0-bin-hadoop2.7/bin/spark-submit --master spark://hadoop-master:7077 --class main.scala.djgarcia.runNoise_IS NoiseIS-1.0-jar-with-dependencies.jar

Save Text File

```
//Write Results
val writer = new
PrintWriter("/home/user/results.txt")
    writer.write(
      "Precision: " + precision + "\n" +
       "Confusion Matrix " + cm + "\n"
    )
writer.close()
```

Outline

- Catching up
- **Introduce Noise**
- Noise Filters
- Instance Selection

Load Train & Test

/opt/spark-2.2.0-bin-hadoop2.7/bin/spark-shell --packages djgarcia:NoiseFramework:1.2,djgarcia:RandomNoise:1.0,djgarcia:SmartReduction:1.0,djgarcia:SmartFiltering:1.0

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.{Vector, Vectors}

//Load Train & Test

val pathTrain = "file:///home/spark/datasets/susy-10k-tra.data"
val rawDataTrain = sc.textFile(pathTrain)

val pathTest = "file:///home/spark/datasets/susy-10k-tst.data"
val rawDataTest = sc.textFile(pathTest)
```

Train and Test RDDs


```
val train = rawDataTrain.map{line =>
   val array = line.split(",")
   var arrayDouble = array.map(f => f.toDouble)
   val featureVector = Vectors.dense(arrayDouble.init)
   val label = arrayDouble.last
   LabeledPoint(label, featureVector)
}

val test = rawDataTest.map { line =>
   val array = line.split(",")
   var arrayDouble = array.map(f => f.toDouble)
   val featureVector = Vectors.dense(arrayDouble.init)
   val label = arrayDouble.last
   LabeledPoint(label, featureVector)
}
```

Cache Train & Test

train.persist

train.count

train.first

test.persist

test.count

test.first

Introduce Noise


```
import org.apache.spark.mllib.util._

val noise = 20 //(in %)

val noisyModel = new RandomNoise(train, noise)

val noisyData = noisyModel.runNoise()

noisyData.persist()

noisyData.count()
```

Decision Tree Clean Dataset


```
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.model.DecisionTreeModel

val numClasses = 2
val categoricalFeaturesInfo = Map[Int, Int]()
val impurity = "gini"
val maxDepth = 20 //Increased Depth
val maxBins = 32

val model = DecisionTree.trainClassifier(train, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)
```

Test Accuracy


```
val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}

val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()

println(s"Test Accuracy Clean Dataset = $testAcc")
```

Test Accuracy Clean Dataset: **0.7058**

Decision Tree Noisy Dataset


```
val model = DecisionTree.trainClassifier(noisyData, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)

val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()

println(s"Test Accuracy Noisy Dataset = $testAcc")
```

Test Accuracy Noisy Dataset: **0.6291**

Outline

- Catching up
- Introduce Noise
- **Noise Filters**
- Instance Selection

Noise Filtering

Available in Spark Packages:

https://spark-packages.org/package/djgarcia/NoiseFramework

NoiseFramework (homepage)

Noise Framework for removing noisy instances with three algorithms: HME-BD, HTE-BD and ENN.

@djgarcia / ☆☆☆☆☆ (2)

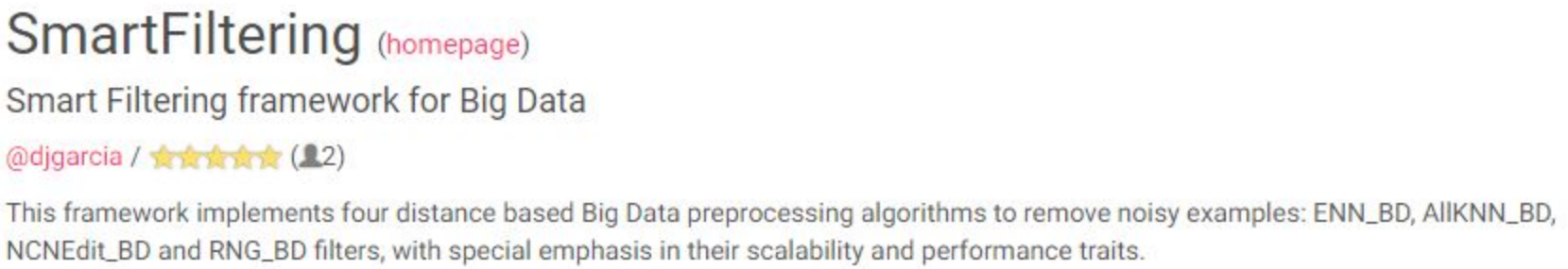
In this framework, two Big Data preprocessing approaches to remove noisy examples are proposed: an homogeneous ensemble (HME_BD) and an heterogeneous ensemble (HTE_BD) filter. A simple filtering approach based on similarities between instances (ENN_BD) is also implemented.

HME_BD, HTE_BD & ENN_BD

Noise Filtering with kNN

Available in Spark Packages:

https://spark-packages.org/package/djgarcia/SmartFiltering



AllKNN_BD, NCNEdit_BD & RNG_BD

HME-BD to Noisy Data


```
import org.apache.spark.mllib.feature._

val nTrees = 100
val maxDepthRF = 10
val partitions = 4

val hme_bd_model = new HME_BD(noisyData, nTrees, partitions,
maxDepthRF, 48151623)

val hme_bd = hme_bd_model.runFilter()

hme_bd.persist()
hme_bd.count()
```

Instances: **6623**

Decision Tree HME-BD


```
val model = DecisionTree.trainClassifier(hme_bd, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy Filtered Dataset= $testAcc")
```

Test Accuracy Filtered Dataset: **0.7927**

HME-BD to Clean Data


```
import org.apache.spark.mllib.feature._

val nTrees = 100
val maxDepthRF = 10
val partitions = 4

val hme_bd_model = new HME_BD(train, nTrees, partitions, maxDepthRF,
48151623)

val hme_bd = hme_bd_model.runFilter()

hme_bd.persist()
hme_bd.count()
```

Instances: **7814**

Decision Tree HME-BD


```
val model = DecisionTree.trainClassifier(hme_bd, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy Filtered Dataset= $testAcc")
```

Test Accuracy Filtered Dataset: **0.7947**

NCNEdit-BD


```
import org.apache.spark.mllib.feature._

val k = 3 //number of neighbors

val ncnedit_bd_model = new NCNEdit_BD(noisyData, k)

val ncnedit_bd = ncnedit_bd_model.runFilter()

ncnedit_bd.persist()

ncnedit_bd.count()
```

Instances: **5821**

Decision Tree NCNEdit-BD


```
val model = DecisionTree.trainClassifier(ncnedit_bd, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy NCNEdit= $testAcc")
```

Test Accuracy NCNEdit: **0.7064**

RNG-BD

```
import org.apache.spark.mllib.feature._

val order = true // Order of the graph (true = first, false = second)
val selType = true // Selection type (true = edition, false = condensation)

val rng_bd_model = new RNG_BD(noisyData, order, selType)
val rng_bd = rng_bd_model.runFilter()

rng_bd.persist()
rng_bd.count()
```

Instances: **7530**

Decision Tree RNG-BD


```
val model = DecisionTree.trainClassifier(rng_bd, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy RNG= $testAcc")
```

Test Accuracy RNG: **0.7052**

Resume

- Original Data:
  - 0.7058
  - 10000
- Original Data with 20% Noise:
  - 0.6291
  - 10000
- HME-BD to Noisy Data / Clean Data:
  - 0.7927 0.7947
  - 6623 7814
- NCNEdit-BD to Noisy Data:
  - 0.7064
  - 5821
- RNG-BD to Noisy Data:
  - 0.7052
  - 7530

Outline

- Catching up
- Introduce Noise
- Noise Filters
- **Instance Selection**

IS with kNN

Available in Spark Packages:

https://spark-packages.org/package/djgarcia/SmartReduction

FCNN_MR, SSMASFLSDE_MR, RMHC_MR & MR_DIS

FCNN-MR


```
import org.apache.spark.mllib.feature._

val k = 3 //number of neighbors

val fcnn_mr_model = new FCNN_MR(train, k)

val fcnn_mr = fcnn_mr_model.runPR()

fcnn_mr.persist()
fcnn_mr.count()
```

Instances: **5584**

Decision Tree FCNN-MR

```
val model = DecisionTree.trainClassifier(fcnn_mr, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy FCNN = $testAcc")
```

Test Accuracy FCNN: **0.6447**

SSMA-SFLSDE-MR


```
import org.apache.spark.mllib.feature._

val ssmasflsde_mr_model = new SSMASFLSDE_MR(train)

val ssmasflsde_mr = ssmasflsde_mr_model.runPR()

ssmasflsde_mr.persist()
ssmasflsde_mr.count()
```

Instances: **187**

Decision Tree SSMA-SFLSDE-MR


```
val model = DecisionTree.trainClassifier(ssmasflsde_mr, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()
println(s"Test Accuracy SSMA-SFLSDE = $testAcc")
```

Test Accuracy SSMA-SFLSDE: **0.7342**

RMHC-MR


```
import org.apache.spark.mllib.feature._

val p = 0.1 // Percentage of instances (max 1.0)
val it = 100 // Number of iterations
val k = 3 // Number of neighbors

val rmhc_mr_model = new RMHC_MR(train, p, it, k, 48151623)
val rmhc_mr = rmhc_mr_model.runPR()

rmhc_mr.persist()
rmhc_mr.count()
```

Instances: **960**

Decision Tree RMHC-MR


```
val model = DecisionTree.trainClassifier(rmhc_mr, numClasses,
categoricalFeaturesInfo, impurity, maxDepth, maxBins)


val labelAndPreds = test.map { point =>
  val prediction = model.predict(point.features)
  (point.label, prediction)
}
val testAcc = 1 - labelAndPreds.filter(r => r._1 != r._2).count().toDouble /
test.count()

println(s"Test Accuracy RMHC = $testAcc")
```

Test Accuracy RMHC: **0.6776**

Resume

- FCNN-MR:
  - 0.7058
  - 5584
  - 44.16%
- SSMA-SFLSDE-MR:
  - 0.7342
  - 187
  - 98.13%
- RMHC-MR:
  - 0.6776
  - 960
  - 90.4%

Clues for Noise and IS

- Distance-based methods (kNN)
- Prepared to work with values in [0,1] range
- Better performance with kNN as classifier

UNIVERSIDAD DE GRANADA

Escuela de Posgrado
Universidad de Granada

MLlib Noise and Instance Selection

Big Data II

Diego García (djgarcia@decsai.ugr.es) 19/04/2018