Tecnológico Nacional De México

Instituto Tecnológico De Tijuana

Subdirección Académica

Departamento de Sistemas y Computación

Semestre Enero - Junio 2022

Ingeniería Informática

Datos Masivos

Práctica 5 - Multilayer Perceptron classifier

Unidad 2

Perez Ortega Victoria Valeria **No.18210718**

Israel López Pablo **No.17210585**

JOSE CHRISTIAN ROMERO HERNANDEZ

Tijuana, B.C. a 18 de Mayo de 2022.

import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

// Load the data stored in LIBSVM format as a DataFrame. - Carga los datos almacenados en formato LIBSVM como DataFrame.

```
scala> import org.apache.spark.ml.classification.MultilayerPerceptronClassifier
import org.apache.spark.ml.classification.MultilayerPerceptronClassifier

scala> import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
```

//val data = spark.read.format("libsvm").load("data/mllib/sample_multiclass_classification_data.txt")
val data = spark.read.format("libsvm").load("C:/Spark/spark-2.4.8-bin-hadoop2.7/data/mllib/sample_multiclass_classification_data.txt")

```
scala> import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

scala> val data = spark.read.format("libsvm").load("C:/Spark/spark-2.4.8-bin-hadoop2.7/data/mllib/sample_multiclass_classificati
")
22/05/17 15:32:44 WARN LibSVMFileFormat: 'numFeatures' option not specified, determining the number of features by going though
If you know the number in advance, please specify it via 'numFeatures' option to avoid the extra scan.
data: org.apache.spark.sql.DataFrame = [label: double, features: vector]
```

// Split the data into train and test - Divide los datos
val splits = data.randomSplit(Array(0.6, 0.4), seed = 1234L)
val train = splits(0)
val test = splits(1)

```
scala> val splits = data.randomSplit(Array(0.6, 0.4), seed = 1234L)
splits: Array[org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]] = Array([label: double, features: vector], [label: double,
vector])

scala> val train = splits(0)
train: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: double, features: vector]

scala> val test = splits(1)
test: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: double, features: vector]
```

// specify layers for the neural network: especificar capas para la red neuronal:

// input layer of size 4 (features), two intermediate of size 5 and 4 capa de entrada de tamano 4 (features), dos intermedias de tamano 5 y 4

// and output of size 3 (classes) y salida de tamano 3 (classes)

val layers = Array[Int](4, 5, 4, 3)

```
scala> val layers = Array[Int](4, 5, 4, 3)
layers: Array[Int] = Array(4, 5, 4, 3)
```

// create the trainer and set its parameters - Crea el trainer y establece sus parametros.

val trainer = new MultilayerPerceptronClassifier()

```
scala> val trainer = new MultilayerPerceptronClassifier()
trainer: org.apache.spark.ml.classification.MultilayerPerceptronClassifier = mlpc_64f15fedf00d
```

.setLayers(layers)

```
scala> .setLayers(layers)
res0: trainer.type = mlpc_64f15fedf00d
```

.setBlockSize(128)

```
scala> .setBlockSize(128)
res1: res0.type = mlpc_64f15fedf00d
```

.setSeed(1234L)

```
scala> .setSeed(1234L)
res2: res1.type = mlpc_64f15fedf00d
```

.setMaxIter(100)

```
scala> .setMaxIter(100)
res3: res2.type = mlpc_64f15fedf00d
```

// train the model * entrena el model

val model = trainer.fit(train)

```
scala> val model = trainer.fit(train)
22/05/17 15:35:19 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
```

// compute accuracy on the test set precision de - calculo en el conjunto de prueba

val result = model.transform(test)

val predictionAndLabels = result.select("prediction", "label")

val evaluator = new MulticlassClassificationEvaluator()

```
scala> val result = model.transform(test)
result: org.apache.spark.sql.DataFrame = [label: double, features: vector ... 3 more fields]

scala> val predictionAndLabels = result.select("prediction", "label")
predictionAndLabels: org.apache.spark.sql.DataFrame = [prediction: double, label: double]

scala> val evaluator = new MulticlassClassificationEvaluator()
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator = mcEval_7d29e02c21f8
```

.setMetricName("accuracy")

```
scala> .setMetricName("accuracy")
res4: evaluator.type = mcEval_7d29e02c21f8
```

println(s"Test set accuracy = ${evaluator.evaluate(predictionAndLabels)}")

```
scala> println(s"Test set accuracy = ${evaluator.evaluate(predictionAndLabels)}")
Test set accuracy = 0.9019607843137255
```