Especialização Lato Sensu em Ciência de Dados e Analytics

Soluções em Processamento para Big Data

{ SPARK Machine Learning Lib }

Prof. Jairson Rodrigues
jairson.rodrigues@univasf.edu.br

{ introdução }

AGENDA

Spark
Data Frames
Resilient Distributed Datasets
MLLib
GraphX

{ spark mllib }

- Machine Learning Library
- Objetivo: tornar a aplicação de algoritmos de aprendizagem de máquina um processo fácil e escalável
- Cobertura:
  - Algoritmos: classificação, regressão, clusterização e filtragem colaborativa
  - Extração de características, transformação de dados e redução de dimensionalidade
  - Pipelines: ferramentas para construir, avaliar e otimizar modelos
  - Persistência: salva e carrega algoritmos e modelos treinados
  - Utilitários: álgebra linear, estatística e manipulação de dados

{ mllib - algoritmos de classificação }

- Regressão Logística
- Árvore de Decisão
- Floresta Randômica
- Gradient-boosted
- Multilayer perceptron
- One-vs-Rest classifier (a.k.a. One-vs-All)
- Naïve Bayes

{ mllib - algoritmos de regressão }

- Regressão Logística
- Árvore de Decisão
- Floresta Randômica
- Gradient-boosted
- Regressão Isotônica

{ mllib - algoritmos de regressão }

- K-means
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)

{ exemplo aplicado }

- Detecção de padrões de ataque em conexões de rede a partir da análise de log conexões disponibilizada pelo MIT Lincoln Labs (KDD Cup 99)
  - https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
- 4.898.431 de observações
- 41 características (sete categóricas, demais contínuas)
- Classe 0 / normal = 972781, 0,19859%,
- Classe 1 / ataque = 3925650, 0,80141%
- 709.18 Megabytes

{ atributos }

Atributo	Descrição	Tipo
duration	length (number of seconds) of the connection	contínua
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discreta
service	network service on the destination, e.g., http, telnet, etc.	discreta
src_bytes	number of data bytes from source to destination	contínua
dst_bytes	number of data bytes from destination to source	contínua
flag	normal or error status of the connection	discreta
land	1 if connection is from/to the same host/port; 0 otherwise	discreta
wrong_fragment	number of ”wrong”fragments	contínua
urgent	number of urgent packets	contínua
hot	number of hot indicators	contínua
num_failed_logins	number of failed login attempts	contínua
logged_in	1 if successfully logged in; 0 otherwise	discreta
num_compromised	number of ”compromised”conditions	contínua
root_shell	1 if root shell is obtained; 0 otherwise	discreta
su_attempted	1 if ”su root”command attempted; 0 otherwise	discreta
num_root	number of ”root”accesses	contínua
num_file_creations	number of file creation operations	contínua
num_shells	number of shell prompts	contínua
num_access_files	number of operations on access control files	contínua
num_outbound_cmds	number of outbound commands in an ftp session	contínua
is_hot_login	1 if the login belongs to the ”hot”list; 0 otherwise	discreta
is_guest_login	1 if the login is a ”guest”login; 0 otherwise	discreta
count	connections to the same host in the past two seconds	contínua
serror_rate	% of connections that have ”SYN”errors	contínua
rerror_rate	% of connections that have ”REJ”errors	contínua
same_srv_rate	% of connections to the same service	contínua
diff_srv_rate	% of connections to different services	contínua
srv_count	connections to the same service in the past two seconds	contínua
srv_serror_rate	% of connections that have ”SYN”errors	contínua
srv_rerror_rate	% of connections that have ”REJ”errors	contínua
srv_diff_host_rate	% of connections to different hosts	contínua
label	0 = normal, 1 = ataque	contínua

{ distribuição dos dados }

Master Name Node

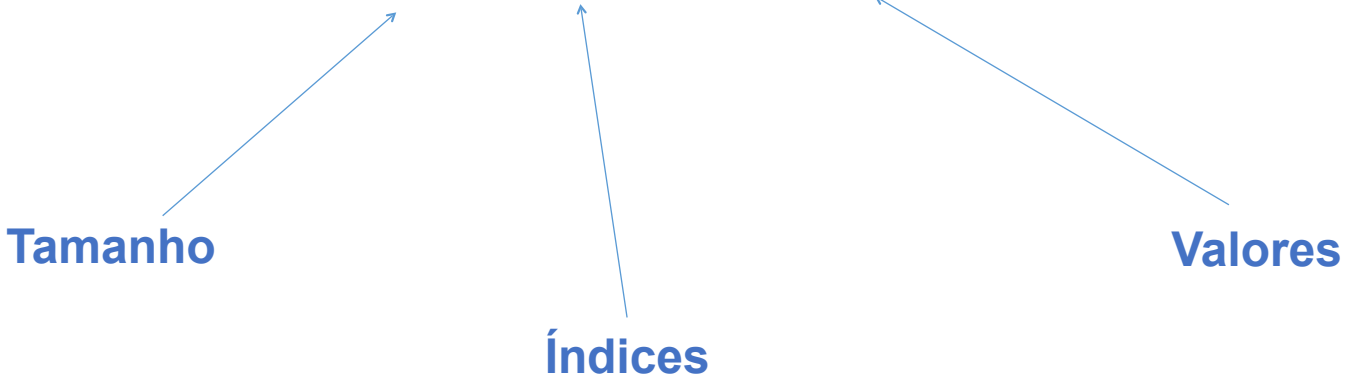
```
ubuntu@name-node:~$ wget http://kdd.org/cupfiles/KDDCupData/1999/kddcup.data.zip
--2016-12-11 21:10:35--  http://kdd.org/cupfiles/KDDCupData/1999/kddcup.data.zip
Resolving kdd.org (kdd.org)... 72.10.51.228
Connecting to kdd.org (kdd.org)|72.10.51.228|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.kdd.org/cupfiles/KDDCupData/1999/kddcup.data.zip [following]
--2016-12-11 21:10:36--  http://www.kdd.org/cupfiles/KDDCupData/1999/kddcup.data.zip
Resolving www.kdd.org (www.kdd.org)... 72.10.51.228
Reusing existing connection to kdd.org:80.
HTTP request sent, awaiting response... 200 OK
Length: 18119640 (17M) [application/zip]
Saving to: 'kddcup.data.zip'

100%[=======================================>] 18,119,640  10.7MB/s   in 1.6s

2016-12-11 21:10:37 (10.7 MB/s) - 'kddcup.data.zip' saved [18119640/18119640]

ubuntu@name-node:~$ unzip kddcup.data.zip
Archive:  kddcup.data.zip
  inflating: kddcup.data.txt
ubuntu@name-node:~$ ls -l kddcup.data.txt
-rw------- 1 ubuntu ubuntu 742579867 Oct 17  2005 kddcup.data.txt
ubuntu@name-node:~$ hdfs dfs -mkdir /kddcup
ubuntu@name-node:~$ hdfs dfs -mkdir /kddcup/input
ubuntu@name-node:~$ hdfs dfs -put kddcup.data.txt /kddcup/input/
ubuntu@name-node:~$
```

{ simplificação pedagógica :-) }

- Todos os tipos de ataque foram classificados como classe = 1
- Padrões que não correspondem a ataques, classe = 0
- Variáveis categóricas foram submetidas a
  - StringIndexer: converte variáveis categóricas para índices
  - OneHotEncoder: converte índices vetores binários
  - VectorAssembler: agrega todas as características em um único vetor esparso

{ vetores esparsos vs vetores densos }

- Representação do resultado em vetores esparsos
- Vetor denso: [1.0, 0.0, 3.0]
- Vetor esparso: (3, [0, 2], [1.0, 3.0]),



{ aplicações auto-contidas }

- Utilizadas para submeter jobs em spark para o cluster

```scala
/* SimpleApp.scala */
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf

object SimpleApp {
  def main(args: Array[String]) {
    val logFile = "YOUR_SPARK_HOME/README.md" // Should be some file on your system
    val conf = new SparkConf().setAppName("Simple Application")
    val sc = new SparkContext(conf)
    val logData = sc.textFile(logFile, 2).cache()
    val numAs = logData.filter(line => line.contains("a")).count()
    val numBs = logData.filter(line => line.contains("b")).count()
    println(s"Lines with a: $numAs, Lines with b: $numBs")
    sc.stop()
  }
}
```

{ aplicações auto-contidas - compilação }

- Em Scala...
- Estrutura do projeto

```
./simple.sbt
./src
./src/main
./src/main/scala
./src/main/scala/SimpleApp.scala
```

Compilação

```
sbt package
...
[info] Packaging {..}/{..}/target/scala-2.11/simple-project_2.11-1.0.jar
```

{ aplicações auto-contidas - dependências }

- Arquivo de dependências (simple.sbt)

```
name := "Simple Project"
version := "1.0"
scalaVersion := "2.11.7"
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.1.0"
```

{ aplicações auto-contidas - execução }

- SPARK-SUBMIT (localhost)

```
YOUR_SPARK_HOME/bin/spark-submit \
  --class "SimpleApp" \
  --master local[*] \
  <path>/target/scala-2.11/simple-project_2.11-1.0.jar
```

- SPARK-SUBMIT (utilizando o cluster em modo YARN)

```
YOUR_SPARK_HOME/bin/spark-submit \
--class "SimpleApp" \
--master yarn \
<path>/target/scala-2.11/kdd-cup_2.11-1.0.jar
```

- SPARK-SUBMIT (utilizando o cluster em modo Standalone)

```
YOUR_SPARK_HOME/bin/spark-submit \
--class "SimpleApp" \
--master spark://spark-node1:7077 \
<path>/target/scala-2.11/kdd-cup_2.11-1.0.jar
```

Maiores informações em: http://spark.apache.org/docs/latest/submitting-applications.html

{ tratamento de dados categóricos }

```
1
2   val indexer: Array[PipelineStage] = originalColumns
3     .map(cname => new StringIndexer()
4       .setInputCol(cname)
5       .setOutputCol(s"${cname}_i")
6       .setHandleInvalid("skip"))
7
8   val one_hot_encoder: Array[PipelineStage] = allIndexedCategoricalColumns
9     .map(cname => new OneHotEncoder()
10      .setInputCol(cname)
11      .setOutputCol(s"${cname}e"))
12
13  val pipelineTmp = new Pipeline().setStages(indexer ++ one_hot_encoder)
14
15  val df = pipelineTmp.fit(textDF).transform(textDF)
16
17  val assembler = new VectorAssembler()
18    .setInputCols(finalFields.diff(Array("label_i")))
19    .setOutputCol("features")
20
21  val outuput = assembler.transform(df).select("label_i", "features")
22    .withColumnRenamed("label_i", "label")
23
24  outuput.write.format("parquet")
25    .save("hdfs://81.14.183.180:9000/kddcup/input/indexed_enconded_data")
```

{ classificador naïve bayes }

```
val random = Math.abs(scala.util.Random.nextInt)
val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3), random)
val bayes = new NaiveBayes()
val pipeline = new Pipeline().setStages(Array(bayes))

val paramGrid = new ParamGridBuilder()
                .addGrid(bayes.smoothing, Array(0.0, 0.1, 0.25, 0.5, 0.75, 0.9, 1.0))
                .build()

val cv = new CrossValidator()
                .setEstimator(pipeline)
                .setEvaluator(new BinaryClassificationEvaluator)
                .setEstimatorParamMaps(paramGrid)
                .setNumFolds(10)

val cvModel = cv.fit(trainingData)

val results = cvModel
                .transform(testData)
                .select("prediction", "label")
```

{ classificador naïve bayes }

```
val rows = results.collect()

val numCorrectPredictions = rows
    .map(row => if (row.getDouble(0) == row.getDouble(1)) 1 else 0)
    .foldLeft(0)(_ + _)

val accuracy = 1.0D * numCorrectPredictions / rows.size

val predictionAndLabels = results
    .rdd.map(x => (x(0).asInstanceOf[Double], x(1).asInstanceOf[Double]))

val metrics = new BinaryClassificationMetrics(predictionAndLabels)
```

Acurácia: de 96%,
Área sob a curva Precision-Recall: 0.917
Área sob a curva ROC: 0.902

{ regressão logística }

```
val random = Math.abs(scala.util.Random.nextInt)
val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3), random)
val lr = new LogisticRegression().setMaxIter(10)
val pipeline = new Pipeline().setStages(Array(lr))

val paramGrid = new ParamGridBuilder()
                .addGrid(lr.regParam, Array(0.1, 0.01))
                .build()

val cv = new CrossValidator()
                .setEstimator(pipeline)
                .setEvaluator(new BinaryClassificationEvaluator)
                .setEstimatorParamMaps(paramGrid)
                .setNumFolds(10)

val cvModel = cv.fit(trainingData)

val results = cvModel
                .transform(testData)
                .select("prediction", "label")
```

{ regressão logística }

```
val rows = results.collect()

val numCorrectPredictions = rows
    .map(row => if (row.getDouble(0) == row.getDouble(1)) 1 else 0)
    .foldLeft(0)(_ + _)

val accuracy = 1.0D * numCorrectPredictions / rows.size

val predictionAndLabels = results
    .rdd.map(x => (x(0).asInstanceOf[Double], x(1).asInstanceOf[Double]))

val metrics = new BinaryClassificationMetrics(predictionAndLabels)
```

Acurácia: de 97,5%,
Área sob a curva Precision-Recall: 0.947
Área sob a curva ROC: 0.945

{ spark-submit }

- KDDCupETL
- KDDCupRL
- KDDCupNaiveBayes

- No **Namenode**

```
spark-submit --class "<nome_do_objeto>" \
--master yarn \
--deploy-mode <modo> \
--conf spark.serializer=org.apache.spark.serializer.KryoSerializer \
/home/ubuntu/mllib/target/scala-2.11/kddcup_2.11-1.0.jar
```

- cluster
- client

{ spark-shell }

```
$ spark-shell
   --master spark://spark-node1:7077
   --num-executors 2
   --driver-memory 6G
   --executor-memory 4G
   --executor-cores 4
   -i KDD-ETL.scala

   … OU KDD-NB.scala
   … OU KDD-LR.scala
```

{ para onde ir agora? }

- Machine Learning Library Guide
- Extração de Características
- Questões de Otimização
- Métricas de Avaliação