AnomalyDetectionDemo

Spark Use Case for Anomaly Detection

Anomaly Detection with Apache Spark - inspired by Sean Owen's presentation https://www.youtube.com/watch?v=TC5cKYBZAeI (https://www.youtube.com/watch?v=TC5cKYBZAeI)

Data available from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html)

Set up

Get data

The unlabeled testdata

```
import sys.process._
"rm -f /tmp/kddcup.data /tmp/kddcup.data.json /tmp/kddcup.testdata.unlabeled /
```

```
val file = "/tmp/kddcup.testdata.unlabeled"
```

```
val dataUrl = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.testdata.unlab

val gz = "/tmp/kddcup.testdata.unlabeled.gz"
s"wget $dataUrl -O $gz"!!

s"gunzip $gz"!!
```

The labeled dataset

```
val origfile = "/tmp/kddcup.data"
```

```
import sys.process._
val dataUrl = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz"

val gz = "/tmp/kddcup.data.gz"
s"wget $dataUrl -O $gz"!!

s"gunzip $gz"!!
```

Create json version of the data

```
val jsonFields = List("duration", "protocol_type", "service", "flag", "src_byt
val jsonFile = origfile + ".json"
```

```
val json = scala.io.Source.fromFile(origfile).getLines.map { line =>
  jsonFields.zip(line.split(","))
            .toMap
            .map { case (k, v) => s""" "$k": "$v" """}
            .mkString("{", ",", "}")
}
val w = new java.io.FileWriter(new java.io.File(jsonFile))
json.foreach { j => w.write(j + "\n") }
w.close
```

Start the server externally on testdata

All the imports

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.mllib.clustering._
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.feature.StandardScaler
import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.rdd._
import org.apache.spark.sql.Row
import org.apache.spark.streaming._
import org.apache.spark.Logging
import org.apache.log4j.{Level, Logger}
```

Reading in and exploring the data -- Spark SQL (and DataFrame)

```
val sqlContext = new SQLContext(sparkContext)
//-rw-r--r--  1 radek  staff   4.3G 15 May 23:44 kddcup.data.json.big
val dataFrame = sqlContext.jsonFile(jsonFile).cache
```

We'll be using the modified (json) training dataset from the competition

Here is an example of the data:

```
"head -n 1 /tmp/kddcup.data.json"!!
```

```
dataFrame.printSchema
```

There are nearly 5 million records

```
dataFrame.count
```

Let's look at the labels

```
val labelsCount = dataFrame.groupBy("label").count().collect
```

```
labelsCount.toList.map( row => (row.getString(0), row.getLong(1)))
```

For simplicity, selecting only non-numeric columns

```
val nonNumericFrame = List("protocol_type" ,"service", "flag")
val labeledNumericFrame = dataFrame.select(
  "label",
  "duration",
  "src_bytes",
  "dst_bytes",
  "land",
  "wrong_fragment",
  "urgent",
  "hot",
  "num_failed_logins",
  "logged_in",
  "num_compromised",
  "root_shell",
  "su_attempted",
  "num_root",
  "num_file_creations",
  "num_shells",
  "num_access_files",
  "num_outbound_cmds",
  "is_host_login",
  "is_guest_login",
  "count",
  "srv_count",
  "serror_rate",
  "srv_serror_rate",
  "rerror_rate",
  "srv_rerror_rate",
  "same_srv_rate",
  "diff_srv_rate",
  "srv_diff_host_rate",
  "dst_host_count",
  "dst_host_srv_count",
  "dst_host_same_srv_rate",
  "dst_host_diff_srv_rate",
  "dst_host_same_src_port_rate",
  "dst_host_srv_diff_host_rate",
  "dst_host_serror_rate",
  "dst_host_srv_serror_rate",
  "dst_host_rerror_rate",
  "dst_host_srv_rerror_rate"
)

labeledNumericFrame.take(1)(0)
```

Running standard SQL queries against your dataset

```
dataFrame.registerTempTable("logs")
val allColumns = sqlContext.sql("SELECT * FROM logs WHERE protocol_type = 'udp
```

Build a clustering model -- Spark MLlib

Prepare the data

Every row becomes (Label, Vector[numeric values])

```
val labeledPoint = labeledNumericFrame.map(row =>
      (row.getString(0), Vectors.dense(row.toSeq.tail.map(s => if(s == null) 0
```

```
val rawData = labeledPoint.map(_._2)
rawData.first
```

Labels are not going to be used when building the model

Scale the features and cache the results

```
val scaler = new StandardScaler().fit(rawData)
```

```
val data = scaler.transform(rawData).cache
```

```
data.first.toArray.toList
```

Use K-Means clustering

```
val numIterations = 10 //in production it should be more
val K = 150
```

```
val clusters = KMeans.train(data, K, numIterations)
```

Meantime let's have a look at Spark UI

Now we have our model, let's apply it to the data

```
// this is a hack to workaround the serialization problems occuring while seri
// we rescope locally everything used by the function to be serialized
// we define the function using the instances
// we launch the computations withing the safe scope
@transient val ser = new java.io.Serializable {
  val lp = labeledPoint
  val cs = clusters
  val sc = scaler
  val f = (x:(String, org.apache.spark.mllib.linalg.Vector)) => (cs.predict(sc
  val predictions = lp.map(x => f(x))
}
```

```
ser.predictions
```

And let's see the clusters and their size

```
val clustersWithSize = ser.predictions.map(x => (x._1, 1)).reduceByKey((x,y) =
```

```
clustersWithSize.take(25).toList
```

Use case assumption:

All the clusters with only one point in them labeled 'normal' are fishy

```
val clustersWithCountAndLabel = clustersWithSize.join(ser.predictions).distinc
clustersWithCountAndLabel.take(20).toList
```

```
//clusters with 1 point and labeled as normal
val suspectedAnomalousClusters = clustersWithCountAndLabel.filter(x => x._2._1
```

Now we have discovered the anomalous clusters

```
val anomalousClusters = suspectedAnomalousClusters.collect
anomalousClusters.toList
```

Listen to the stream of events and predict anomaly -- Spark streaming

To create the real-time stream we will use the test dataset from the competition (in CSV format)

```
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,107,6,0.00,0.00,1.00,1.00
0.06,0.07,0.00,255,6,0.02,0.05,0.00,0.00,0.00,0.00,1.00,1.00
```

We will use a simple Java app that reads in the logs and sends them to a TCP socket

Create streaming context with batch 2s

```
val ssc = new StreamingContext(sparkContext, Seconds(2))
```

Turn down the logging

```
Logger.getRootLogger.setLevel(Level.ERROR)
```

Helper method for removing non-numeric columns

```
def extractNumericColumns(r: RDD[Array[String]]): RDD[Vector] = {
  val nonNumericColumns = List(1, 2, 3)
  r.map(s => Vectors.dense(s.filterNot(f => nonNumericColumns.contains(s.index
                           map(st => if(st == null) 0.0 else st.toDouble).toAr
}
```

Helper method for finding the anomalies

```
def findAnomaly(r: RDD[Int], anomalousClusters: Array[Int]): RDD[String] = {
  r.filter(x => anomalousClusters.contains(x)).map(x => "Suspected anomaly - e
}
```

Start listening to the stream

```
val lines = ssc.socketTextStream("localhost", 9999)
```

```
@transient val safeScope = new java.io.Serializable {
  val cs = clusters
  val sc = scaler
  val ac = anomalousClusters
  val compute = lines .map(x => x.split(","))
                      .transform(y => extractNumericColumns(y))
                      .transform(x => cs.predict(sc.transform(x)))
                      .transform(x => findAnomaly(x, ac)).print

}
```

Hackers and fraudsters beware!

```
ssc.start()
ssc.awaitTermination()
```

Spark UI now is showing Streaming tab

```
ssc.stop()
```

Build: | **buildTime**-*Sat Jan 09 20:28:53 UTC 2016* | **formattedShaVersion**-*0.6.2-7c7b07797474ce69a7edeee78cd1c1df09bd5730* | **sbtVersion**-*0.13.8* | **scalaVersion**-*2.11.7* | **sparkNotebookVersion**-*0.6.2* | **hadoopVersion**-*2.7.1* | **jets3tVersion**-*0.7.1* | **jlineDef**-*(jline,2.12)* | **sparkVersion**-*1.6.0* | **withHive**-*true* | **withParquet**-*true* |.