



INSTALLING AND RUNNING SPARK

First of all

2

- ❑ **Virtual Machine with all that you would need**

VMWare Player:



https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/14_0

User: spark | Pass: spark | RootPass: spark

<https://mega.nz/#!DdRnhAYJ!7PeO-U28OAadLb8KFYBsa0CDSSX74t9c6QdqWmEdA2A>

Outline

3

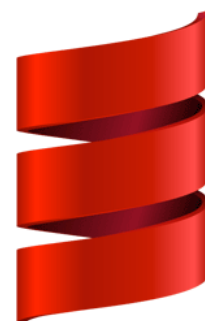
- **Downloading and Installing Spark**
- **Datasets: SUSY**
- **Datasets: ECBDL_14**
- **MLlib: Machine Learning on Spark**
 - ▣ **Statistical**
 - ▣ **Spark-packages**
 - ▣ **kNN-IS**

Downloading and Installing Spark

4

Scala

- ❑ `wget http://scala-lang.org/files/archive/scala-2.11.6.tgz`
- ❑ `tar xzf scala-2.11.6.tgz -C /usr/local`
- ❑ `cd /usr/local/`
- ❑ `ln -s scala-2.11.6 scala`
- ❑ `vi /etc/profile.d/scala.sh`



Scala

```
export SCALA_HOME=/usr/local/scala
```

```
export PATH=${SCALA_HOME}/bin:${PATH}
```

Downloading and Installing Spark

5

Maven

- ❑ `wget http://mirror.cc.columbia.edu/pub/software/apache/maven/maven-3/3.0.5/binaries/apache-maven-3.0.5-bin.tar.gz`

- ❑ `tar xzf apache-maven-3.0.5-bin.tar.gz -C /opt`

- ❑ `mv /opt/apache-maven-3.0.5/ /opt/maven`

- ❑ `vi /etc/profile.d/maven.sh`

maven

```
export M2_HOME=/opt/maven
```

```
export PATH=${M2_HOME}/bin:${PATH}
```

Downloading and Installing Spark

6

Spark

- ❑ `wget http://ftp.cixug.es/apache/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2.7.tgz`
- ❑ `tar xzf spark-2.2.0-bin-hadoop2.7.tgz -C /opt/`
- ❑ `vi /etc/profile.d/spark.sh`
`export SPARK_HOME=/opt/spark-2.2.0-bin-hadoop2.7`
`export PATH=${SPARK_HOME}/bin:${PATH}`



Downloading and Installing Spark

7

Scala IDE

- ❑ `wget http://downloads.typesafe.com/scalaide-pack/4.7.0-vfinal-oxygen-212-20170929/scala-SDK-4.7.0-vfinal-2.12-linux.gtk.x86_64.tar.gz`
- ❑ `tar -xvzf scala-SDK-4.7.0-vfinal-2.12-linux.gtk.x86_64 -C /opt`
- ❑ `chmod -R +r /opt/eclipse/`
- ❑ `vi /usr/bin/eclipse`

```
#!/bin/sh
export ECLIPSE_HOME="/opt/eclipse"
$ECLIPSE_HOME/eclipse $*
```
- ❑ `chmod 755 /usr/bin/eclipse`



Downloading and Installing Spark

8

Scala IDE

□ vi /usr/share/applications/eclipse.desktop

[Desktop Entry]

Encoding=UTF-8

Name=Eclipse

Comment=Eclipse Mars.1 4.5.1

Exec=eclipse

Icon=/opt/eclipse/icon.xpm

Terminal=false

Type=Application

Categories=GNOME;Application;Development;

StartupNotify=true



Downloading and Installing Spark

9

Update Path

- ☐ `source /etc/profile.d/*.sh`
- ☐ Log Out & Log In
- ☐ Restart

Outline

10

- Downloading and Installing Spark
- **Datasets: SUSY**
- Datasets: ECBDL_14
- MLlib: Machine Learning on Spark
 - ▣ Statistical
 - ▣ Spark-packages
 - ▣ kNN-IS

Dataset: SUSY

11

SUSY

- Subset of 10,000 instances in the VM. (5M original)
- Number of instances: 10.000 + 10.000 (18 features)
 training test
- Classification problem
- Real features

Dataset: SUSY

12

SUSY: Path

□ Local (Virtual Machine):

- ▣ Training: `/home/spark/datasets/susy-10k-tra.data`
- ▣ Test: `/home/spark/datasets/susy-10k-tst.data`
- ▣ Header: `/home/spark/datasets/susy.header`

Outline

13

- Downloading and Installing Spark
- Datasets: SUSY
- **Datasets: ECBDL_14**
- MLlib: Machine Learning on Spark
 - ▣ Statistical
 - ▣ Spark-packages
 - ▣ kNN-IS

- Arnaldo, K. Veeramachaneni, U. O'Reilly and J. Bacardit. *ECBDL14 dataset: Protein structure prediction and contact map for the ECBDL2014 big data competition*, 2014, [Link](#)

Dataset: ECBDL_14

15

ECBDL_14: Path

□ Cluster (HDFS):

- Training: `/user/spark/datasets/ECBDL14_mbd/ecbd14tra.data`
- Test: `/user/spark/datasets/ECBDL14_mbd/ecbd14tst.data`
- Header: `/user/spark/datasets/ECBDL14_mbd/ecbd14.header`

Outline

16

- Downloading and Installing Spark
- Datasets: SUSY
- Datasets: ECBDL_14
- **MLlib: Machine Learning on Spark**
 - Statistical
 - Spark-packages
 - kNN-IS

Mllib: Running

17

Virtual Machine examples

Interactive Shell Scala (Virtual Machine):

- ❑ `/opt/spark-2.2.0-bin-hadoop2.7/bin/spark-shell`

Spark Submit (Virtual Machine):

- ❑ `/opt/spark-2.2.0-bin-hadoop2.7/bin/spark-submit`

Interactive Shell R (Virtual Machine):

- ❑ `/opt/spark-2.2.0-bin-hadoop2.7/bin/sparkR`

Mllib: Running

18

Cluster examples

Interactive Shell Scala (Virtual Machine):

- `/opt/spark-2.2.0/bin/spark-shell`

Spark Submit (Virtual Machine):

- `/opt/spark-2.2.0/bin/spark-submit`

Interactive Shell R (Virtual Machine):

- `/opt/spark-2.2.0/bin/sparkR`

Outline

19

- Downloading and Installing Spark
- Datasets: SUSY
- Datasets: ECBDL_14
- MLlib: Machine Learning on Spark
 - ▣ **Statistical**
 - ▣ Spark-packages
 - ▣ kNN-IS

Mllib with Scala: Statistical



20

Basic Statistics



- Summary statistics
- Correlations
- Stratified sampling
- Hypothesis testing
 - Streaming Significance Testing
- Random data generation

<https://github.com/JMailloH/BasicStatisticsSpark>

<https://spark.apache.org/docs/2.2.0/mllib-statistics.html#summary-statistics>

Statistical: import

21

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.stat.{ MultivariateStatisticalSummary, Statistics }
import scala.collection.mutable.ListBuffer
```

Statistical: Read

22

```
val train = sc.textFile(pathTrain).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)
}.persist

val test = sc.textFile(pathTest).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)
}.persist
```

Statistical: StatisticalSummary

23

```
val observationsTrain = train.map(_.features)
```

```
val observationsTest = test.map(_.features)
```

```
val summaryTrain: MultivariateStatisticalSummary = Statistics.colStats(observationsTrain)
```

```
val summaryTest: MultivariateStatisticalSummary = Statistics.colStats(observationsTest)
```

summaryTrain.max is a dense vector containing the max value for each column

Statistical: StatisticalSummary

24

```
val observationsTrain = train.map(_.features)
```

```
val observationsTest = test.map(_.features)
```

```
val summaryTrain: MultivariateStatisticalSummary = Statistics.colStats(observationsTrain)
```

```
val summaryTest: MultivariateStatisticalSummary = Statistics.colStats(observationsTest)
```

summaryTrain.max is a dense vector containing the max value for each column

Statistical: Result and write

25

```
var outputString = new ListBuffer[String]

outputString += "*****TRAIN*****\n\n"

outputString += "@Max (0) --> " + summaryTrain.max(0) + "\n"
outputString += "@Min (0) --> " + summaryTrain.min(0) + "\n"
outputString += "@Mean (0) --> " + summaryTrain.mean(0) + "\n"
outputString += "@Variance (0) --> " + summaryTrain.variance(0) + "\n"
outputString += "@NumNonZeros (0) --> " + summaryTrain.numNonzeros(0) + "\n"

outputString += "\n\n*****TEST*****\n\n"

outputString += "@Max (0) --> " + summaryTest.max(0) + "\n"
outputString += "@Min (0) --> " + summaryTest.min(0) + "\n"
outputString += "@Mean (0) --> " + summaryTest.mean(0) + "\n"
outputString += "@Variance (0) --> " + summaryTest.variance(0) + "\n"
outputString += "@NumNonZeros (0) --> " + summaryTest.numNonzeros(0) + "\n"
```

Statistical: StatisticalSummary

26

```
val predictionsTxt = sc.parallelize(outputString, 1)
```

```
predictionsTxt.saveAsTextFile(pathOutput)
```

Number of partition = 1 to obtain only 1 part

SaveAsTextFile for readable stile

Statistical: compile.sh & clean.sh

27

src/main/scala

LICENSE

README.md

clean.sh

compile.sh

kNN_IS.scala

pom.xml

run.sh

Build the .jar file

```
mvn package -Dmaven.test.skip=true
```

Remove old compilation

```
mvn clean
```


Run the example with

```
./clean.sh ; ./compile.sh ; ./run.sh
```



Statistical: run.sh

28

 src/main/scala LICENSE README.md clean.sh compile.sh kNN_IS.scala pom.xml run.sh

Submit an application

```
/opt/spark-2.2.0-bin-hadoop2.7/bin/spark-submit  
--master local[*]  
--class org.apache.spark.run.runBasicStatistics  
./target/BasicStatistics-1.0.jar  
/home/spark/datasets/susy-10k-tra.data  
/home/spark/datasets/susy-10k-tst.data  
./outputBasicStatistics
```



Outline

29

- Downloading and Installing Spark
- Datasets: SUSY
- Datasets: ECBDL_14
- MLlib: Machine Learning on Spark
 - ▣ Statistical
 - ▣ **Spark-packages**
 - ▣ kNN-IS

Spark-packages

30

<https://spark-packages.org/>

SparkPackages

Feedback

Register a package

Login

[Find a package](#)

A community index of third-party packages for **Apache Spark**.

Showing packages 1 - 50 out of 394

[Next >](#)

All (394)	Core (14)	Data Sources (48)	Machine Learning (79)	Streaming (54)	Graph (18)	PySpark (17)	Applications (14)	Deployment (12)	Examples (24)	Tools (30)
--------------	--------------	-------------------------	--------------------------	-------------------	---------------	-----------------	----------------------	--------------------	------------------	---------------

- ▣ Large number of "utils" implementations
- ▣ Machine learning algorithms
- ▣ Many other third parties algorithm for spark

Spark-packages

31

<https://spark-packages.org/package/h2oai/sparkling-water>

sparkling-water (homepage)

Sparkling Water provides H2O algorithms inside Spark cluster

@h2oai / ★★★★★ (2)

Sparkling Water connects worlds of H2O and Spark. It enables launching H2O on top of Spark and using H2O capabilities including various ML algorithms, graphical user interface, or R integration.

Spark-packages

32

<https://spark-packages.org/package/h2oai/sparkling-water>

sparkling-water (homepage)

Sparkling Water provides H2O algorithms inside Spark cluster

@h2oai / ★★★★★ (2)

Sparkling Water connects worlds of H2O and Spark. It enables launching H2O on top of Spark and using H2O capabilities including various ML algorithms, graphical user interface, or R integration.



Michal Malohlava • 3 years ago

More about Sparkling Water is available at h2o blog: <http://h2o.ai/blog>

Example code is part of GitHub repository: <https://github.com/h2oai/sp...>

^ | v • Reply • Share ›

Spark-packages

33

<https://spark-packages.org/package/lensacom/sparkit-learn>

sparkit-learn (homepage)

PySpark + Scikit-learn = Sparkit-learn

@lensacom / ★★★★★ (2)

Sparkit-learn aims to provide scikit-learn functionality and API on PySpark. The main goal of the library is to create an API that stays close to sklearn's.

The driving principle was to "Think locally, execute distributively." To accomodate this concept, the basic data block is always an array or a (sparse) matrix and the operations are executed on block level.

Spark-packages

34

<https://spark-packages.org/package/lensacom/sparkit-learn>

sparkit-learn (homepage)

PySpark + Scikit-learn = Sparkit-learn

@lensacom / ★★★★★ (2)

Sparkit-learn aims to provide scikit-learn functionality and API on PySpark. The main goal of the library is to create an API that stays close to sklearn's.

The driving principle was to "Think locally, execute distributively." To accomodate this concept, the basic data block is always an array or a (sparse) matrix and the operations are executed on block level.

How to

This package doesn't have any releases published in the Spark Packages repo, or with maven coordinates supplied. You may have to build this package from source, or it may simply be a script. To use this Spark Package, please follow the instructions in the README.

Spark-packages

35

<https://spark-packages.org/package/Azure/mmlspark>

mmlspark (homepage)

Microsoft Machine Learning for Apache Spark

@Azure / ★★★★★ (13)

MMLSpark provides a number of deep learning and data science tools for Apache Spark, including seamless integration of Spark Machine Learning pipelines with Microsoft Cognitive Toolkit (CNTK) and OpenCV, enabling you to quickly create powerful, highly-scalable predictive and analytical models for large image and text datasets.

Tags

3 ml 3 machine learning 3 Microsoft 3 Azure 3 pyspark 3 cntk 3 Microsoft Machine Learning 2 tools

How to [+]

Include this package in your Spark Applications using:

spark-shell, pyspark, or spark-submit

```
> $SPARK_HOME/bin/spark-shell --packages Azure:mmlspark:0.11
```

```
$SPARK_HOME/bin/spark-shell --packages Azure:mmlspark:0.11
```

Outline

36

- Downloading and Installing Spark
- Datasets: SUSY
- Datasets: ECBDL_14
- MLlib: Machine Learning on Spark
 - ▣ Statistical
 - ▣ Spark-packages
 - ▣ **kNN-IS**

Mllib: Spark-packages

37

https://spark-packages.org/package/JMailloH/kNN_IS



kNN_IS (homepage)

kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data.

@JMailloH / ★★★★★ (4)

This is an open-source Spark package about an exact k-nearest neighbors classification based on Apache Spark. We take advantage of its in-memory operations to simultaneously classify big amounts of unseen cases against a big training dataset. The map phase computes the k-nearest neighbors in different splits of the training data. Afterwards, multiple reducers process the definitive neighbors from the list obtained in the map phase. The key point of this proposal lies on the management of the test set, maintaining it in memory when it is possible. Otherwise, this is split into a minimum number of pieces, applying a MapReduce per chunk, using the caching skills of Spark to reuse the previously partitioned training set.

```
>$: /opt/spark-2.2.0-bin-hadoop2.7/bin/spark-shell  
--packages JMailloH:kNN_IS:3.0
```

Mllib: Script with kNN-IS

38

https://spark-packages.org/package/JMailloH/kNN_IS

src/main/scala

LICENSE

README.md

clean.sh

compile.sh

kNN_IS.scala

pom.xml

run.sh



Ctrl

+ C

Ctrl

+ V



kNN-IS: import

39

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.spark.rdd._
import org.apache.spark.mllib.linalg.Vectors
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.evaluation.MulticlassMetrics
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.classification.kNN_IS.kNN_IS
import utils.keel.KeelParser
import scala.collection.mutable.ListBuffer
```

kNN-IS: Read

40

```
val converter = new KeelParser(sc, "/home/hadoop/datasets/susy.header")
```

```
val train = sc.textFile("/home/hadoop/datasets/susy-10k-tra.data", 10).  
  map(line => converter.parserToLabeledPoint(line)).persist
```

```
val test = sc.textFile("/home/hadoop/datasets/susy-10k-tst.data", 10).  
  map(line => converter.parserToLabeledPoint(line)).persist
```


kNN-IS: Param.

41

```
val k = 5
```

```
val dist = 2 //euclidean
```

```
val numClass = converter.getNumClassFromHeader()
```

```
val numFeatures = converter.getNumFeaturesFromHeader()
```

```
val numPartitionMap = 10
```

```
val numReduces = 2
```

```
val numIterations = 1
```

```
val maxWeight = 5
```

```
val knn = kNN_IS.setup(train, test, k, dist, numClass, numFeatures,  
numPartitionMap, numReduces, numIterations, maxWeight)
```

kNN-IS: Classify

42

```
val predictions = knn.predict(sc)
```

```
val metrics = new MulticlassMetrics(predictions)
```

```
val precision = metrics.precision
```

```
val cm = metrics.confusionMatrix
```

```
val binaryMetrics = new BinaryClassificationMetrics(predictions)
```

```
val AUC = binaryMetrics.areaUnderROC
```



INSTALLING AND RUNNING SPARK