An introduction to statistical computing in Scala

Roxana Tesileanu

INCDS 2017

Table of Contents

[Getting started 1](#__RefHeading___Toc214_241067100)

[Basic vector operations 2](#__RefHeading___Toc129_1943657966)

[Reading files for classification tasks 3](#__RefHeading___Toc809_1943657966)

[K-Nearest Neighbors (kNN) classification 4](#__RefHeading___Toc1178_1943657966)

[Bibliography 4](#__RefHeading___Toc130_537809076)

# Getting started

This work gathers together my learning steps towards using Scala as an environment for statistical computing. My background in statistics and linear algebra helped a lot, so it does assume the interested people passed their statistics courses. The big advantage of Scala is it allows you to express your thoughts in a natural manner, so having a clear idea of what you want to implement really helps a lot.

In order to begin with statistical computing in Scala, you need to install the appropriate tools on your computer. Because different operating systems require different installing procedures, I will let you check the websites of the tools used and extract the information needed to install them on your computer. So, visit the website of the Scala language and of sbt and follow the instructions found there. If you encounter big problems you can grasp to the introduction into the Scala language of Jason Swartz1 and come back later. This is a normal iterative approach used by anyone who wants to get into a new field. The main aim of this work is to serve as documentation for the scalaML package developed by me at INCDS. The GitHub repository of the project can be found at the following link:

[https://github.com/RoxanaTesileanu/multivariate\_analyses/tree/master/DeepLearning/src/main/scala/com/mai/scalaML](https://github.com/RoxanaTesileanu/multivariate_analyses/tree/master/DeepLearning/src/main/scala/com/incds/scalaML).

As mentioned previously, in order to use the scalaML package, you need to install Scala and sbt. Supposed you passed through these initial steps successfully, you can create a new Scala project using sbt. Again, if you encounter big problems, you can check the book of Joshua Suereth and Matthew Farwell2 for an introduction in sbt.

# Basic vector operations

Because most statistical algorithms manipulate datasets which are collections of vectors, manipulating vectors is an essential task. I’ve chosen to implement some of the most widely used vector operations in Scala, because I try to stick to the main Scala types as much as possible while developing ML algorithms. This makes the further development task easier. The vector operations are found in the BasicVectorOP.scala file of the scalaML package and include: vector addition, vector subtraction, elementwise multiplication and dot product for any two vectors of type Array[Double], and matrix multiplication for any two matrices of type Array[Array[Double]], where each Array[Double] represents an observation (i.e. a row of the dataset) and all elements with the same index of all the inside arrays (i.e. of any Array[Double]) represent a column (i.e. a variable of the dataset).

## Vector addition

Suppose you have two vectors a and b of type Array[Double]. You can perform an elementwise addition with the function arrayAdd(a,b). Suppose you’ve started a Scala session with sbt. You can test the function in REPL using the following commands:

scala> import com.mai.scalaML.BasicVectorOP.\_

import com.mai.scalaML.BasicVectorOP.\_

scala> val a = Array(1.0, 2.0, 1.0)

a: Array[Double] = Array(1.0, 2.0, 1.0)

scala> val b = Array(2.0, 0.0, 1.0)

b: Array[Double] = Array(2.0, 0.0, 1.0)

scala> arrayAdd(a,b)

res0: Array[Double] = Array(3.0, 2.0, 2.0)

## Vector subtraction

Using the same two vectors from above you can test the arraySubt() function:

scala> arraySubt(a,b)

res1: Array[Double] = Array(-1.0, 2.0, 0.0)

## Elementwise multiplication of two vectors

Using the same two vectors from above you can test the arrayMultipl() function:

scala> arrayMultipl(a,b)

res2: Array[Double] = Array(2.0, 0.0, 1.0)

## Dot product of two vectors

Using the same two vectors from above you can test the dot() function:

scala> dot(a,b)

res3: Double = 3.0

## Matrix multiplication

Suppose you have two matrices a and b of type Array[Array[Double]]. You can test the matrixMultipl() in REPL:

scala> val a = Array(Array(0.1, 0.2), Array(0.1, 0.2))

a: Array[Array[Double]] = Array(Array(0.1, 0.2), Array(0.1, 0.2))

scala> val b = Array(Array(0.3, 0.4), Array(0.5, 0.6))

b: Array[Array[Double]] = Array(Array(0.3, 0.4), Array(0.5, 0.6))

scala> matrixMultipl(a,b)

res4: Array[Array[Double]] = Array(Array(0.13, 0.16), Array(0.13, 0.16))

# Reading files for classification tasks

You can read txt and csv files for classification tasks using the readFileClassif() function found in the ReadFile object. To use it import the ReadFile object in REPL. The first row of the file to be read is assumed to contain the header and the last column is assumed to contain the labels. The function returns a tuple with the information needed for classification (i.e. a dataMatrix, the dataLabels and the used classes).

scala> import com.mai.scalaML.ReadFile.\_

import com.mai.scalaML.ReadFile.\_

scala> val path = "path to file”

path: String = path to file

scala> val myData = readFileClassif(path, "\t")

myData: (Vector[Array[Double]], Vector[Int], Range) = (Vector(Array(14488.0, 7.153469, 1.673904), Array(26052.0, 1.441871, 0.805124), …), Vector(2, 1, 1, 1, 1, 3, 3, 1, 3, 1,……), Range(1,2,3))

The readFileClassif() function, takes two parameters of type String: the path to the file and the delimitation of the file (i.e. “,” for comma delimited files, or “\t” for tab delimited files). The variable val mydata is the return of the function and represents a tuple of three pieces of information which can be accessed using the tuple indexes :

scala> myData.\_1

res6: Vector[Array[Double]] = Vector(Array(14488.0, 7.153469, 1.673904), Array(26052.0, 1.441871, 0.805124),……

scala> myData.\_2

res7: Vector[Int] = Vector(2, 1, 1, 1, 1, 3, 3, 1, 3, 1, 1, 2, 1, 1, 1, 1, 1, 2, 3, 2, 1, 2, 3, 2, 3, 2, 3, 2, 1, 3, 1, 3, 1, 2, 1, 1,……

scala> myData.\_3

res8: Range = Range(1, 2, 3)

The above pieces of information represent the input for the classification algorithms: a Vector[Array[Double]] which represents the data matrix, a Vector[Int] which represents the labels and a Range which stores the used classes.

# K-Nearest Neighbors (kNN) classification

The object kNN found in the kNN.scala file contains the necessary items for performing a kNN classification.

# Bibliography

1. Swartz, J. *Learning Scala*. (O’Reilly, 2015).

2. Suereth, J. & Farwell, M. *SBT in action: the simple Scala Build Tool*. (Manning Publications Co., 2016).