Learn you some

for greater good!

Luke Geeson
A modification of a talk done for HackSoc by Luke Geeson
adapted for G53MLE (non-examinable)

What's up?

- An intro to Scala: A Scalable language used a lot in industry for parallelising computation

- An intro to Spark: An open source data-mining framework for massive scale data-mining and machine learning

- How both apply to Machine learning

But what is Scala?

Scala: The *scalable* language

Scala = Java + Functional Programming

Makes working concise, more so than java!

widely used in industry!

Why Scala?

- Runs on the Java JVM, so Java code and Scala can be run on the same stack. Compiles to Java Bytecode, so its pretty portable!
- Java and Scala are interoperable
- Static type system and automatic type inference
- in built asynchronous data handling and parallelisation (using Java-style futures and promises), oh and lazy evaluation! = scalability!
- Pattern matching: switch statements on steroids!
- Higher order functions and functional programming == expressibility

The history

- Started in 2001, by Martin Odersky, following work on funnel (another functional language)
- released publicly in 2004, on the java platform
- in 2011, Scala received €2.3 million from the European Research Council, allowing it to get commercial support
- used in industry a lot today!

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

A String

Hello, world!

```
object HelloWorld{
    def main(args: Array[String]) {
        println("Hello, world!")
    }
}
```

A function

A String

Hello, world!

A class/Object (this one is technically a ‘singleton’ object)

```
object HelloWorld{
    def main(args: Array[String]) {
A function ⟶  println("Hello, world!")
    }
}
```

A String

Files and some admin

Scala files are saved with .scala extensions,

e.g. helloWorld.scala

once Scala is installed, it comes with a ‘compiler’ to turn your .scala files into runnable byte code for the machine

this compiler is called scalac

compile your programs with: scalac filename.scala

run with scala filename

e.g. scala HelloWorld

Scala’s compilation model is identical to Java, and so you can use it with build systems like Ant or Gradle

Demo

Comments

- Comments are the same in java:

```
// for single line comments

/*
   for block comments on multiple lines
*/
```

Variable Declaration

- Declare that variables exist with either `var` or `val`
- variables declared with `val` are **immutable**, that is they cannot change once set (like final in java)
- variables declared with `var` are **mutable,** for the java fans out there
- good practice to use `val` where you can

```
val num = 10
num = 20                        //error: reassignment to val
var meaningOfLife = 42
meaningOfLife = 34              //this is fine
meaningOfLife = 3.1415          //this is not, why?
```

Are you my type?

- Scala is **statically** typed, but it uses **type inference** too!

- static typing means it checks the variable types at **compilation** time so it doesn't have to when it runs (a bit like java, but not like python)

- Type inference means it automatically **guesses** the **types** of your expressions/statements and matches them accordingly, or **moans** at you if they are wrong.

  you can manually specify the type however with colons:

```
val numTriforceTriangles : Int = 3

val radius : Double = 33  //automatically converts types like these
```

some more types

- Spark has lots of inbuilt types, here are some more:

Int

Double

Boolean

Char

- Scala has a LOT of type stuff related to functional programming, if you're interested: https://twitter.github.io/scala_school/type-basics.html

Java is Spark is Java

- All of the standard class methods, and classes that come with **Java** can also be **directly used with Scala**, e.g the String class (and the syntax is the same):

```
"hello world".length()
"yolo swaggins".substring(4, 8)
```

- There is also some Scala specific methods, which are **functional** in nature

```
"hello world".take(4)  //takes the first 4 characters from the string
```

- See official Scala documentation for more

this is fun

- define functions like so:

```
def functionName(args...): ReturnType = { body... }

def fizzBuzz(x:Int){
   if (x % 3 == 0)
      println("fizz")
   else if (x % 5 == 0)
      println("buzz")
   else
      println(x)
}
```

- the **last** expression in the function block is the **return** value
- can **omit** the {} for the **function** block or **if** statements if they are **single** statements
- invoking functions is the **same** as in **every** language

more function stuff

- Give your arguments **default** values with '=':
  ```
  def defaultsInMyFunc(x : Int = 4) = {...}
  ```

- Make **anonymous** functions like so:
  ```
  (number : Int) => number + 1
  ```

- **or** like so:
  ```
  val incr : Int => Int = _ + 1
  ```

- check out https://www.coursera.org/course/progfun
  if you want to know about some crazy **functional** stuff

go with the flow

- If, while and do-while statements are the same as Java and C++

- you can specify a **range** of values using ‘to’

```
1 to 5
```

- you can **cycle** over these ranges using .foreach

```
(1 to 5).foreach(
   (number : Int) => println(number +1)
)
```

There’s more!

- Scala has **lots** of functional programming stuff, it has full support for **object oriented** programming too, I won’t go into detail as this could be a whole module in itself!
  checkout the resources!

- ON TO SPARK

A “fast and general engine for large-scale data processing”

part of the apache open source project

Great for large-scale, cluster based computing

also has a substantial Machine Learning component for large scale parallelised model training, prediction and use

but what Sparks your interest?

- Allows you to take advantage of lots of computers interconnected to do some big number crunching

- can scale with thousands of nodes in a cluster, and has been proven to run pretty fast too!

- can be used with Java, Python, Scala (the main one) and R

- It's pretty generalised, but also has some complex in built analytics and machine learning packages

- Also OPEN SOURCE

So shiny, the Shell Sparks

- Once you download apache spark, assuming you have Scala and Java, you can run spark from ./bin/spark-shell with an interactive shell

- you can then input some Scala or use some of the Spark specific stuff if you have some data

Fundamentals

- Spark has a fundamental data type called an RDD object. An RDD object represents the input data as rows. Invoke one as follows:

```
val textFile = sc.textFile("testData.txt")
//sc is an in built type for a SparkContext
textFile.count() //returns number of rows
textFile.distinct() //returns distinct rows
textFile.first() //grabs the first item only
textFile.isEmpty() //returns true if empty
//and so on...
```

Combine with Spark Functions for more jazz

- We can do use Scala's functional programming style to nicely do some number crunching in a clear way:

```
val textFile = sc.textFile("shakespear.txt")
val linesWithWord =
textFile.filter(line => line.contains("the"))

println(linesWithWord.count())
```

The Science of it all

- Why not use a local machine with in built commands to do this? It's simple, most in built programs don't parallelise

- Spark is designed to run across many hundreds of servers, in what are known as clusters, and maybe even across sites around the world.

- All of the concurrency, parallelism and merging of data is handled by spark! provided you have a cluster and know how to set it up.

Cluster-one

- **Big** Systems in the world run in DataCentres
- DataCentres contain hundreds of racks
- racks contain (sometimes) hundreds of nodes
- nodes are computers connected up together (via switches) and are often controlled by one central master node (another computer)
- Associated nodes all (generally) run the same software and are grouped (read clustered) together to perform parallelised computations on a massive data set
- The master node issues commands to the ‘slave’ nodes, the slave nodes then take a portion of the data each and do their part. Finally, the result of the computations from all nodes are combined into one result and returned to the master node. This is the basis of a DataMining Algorithm called MapReduce.

MapReduce

- It's Redundant! As the data is distributed across multiple nodes (DFS), and processed across multiple nodes, it has loads of failure tolerance. When a server is used to just store data, it is known as a chunk server

- It's convenient, chunk servers are clustered together and can be repurposed to do computations (these are compute servers) and they can use their local store of the data to do computation. This can be done remotely

- It is abstract, This paradigm is a well established way of doing things, lot's of libraries support this idea without the need for you to directly access the file system

No More theory, just code!

- val wordCounts = textFile.flatMap(line => line.split(" “))

  //takes each line and splits it by an empty string

No More theory, just code!

- `val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1))`

//takes each list of words from a line, and pairs them up in a tuple (a collection of variables) with a number

No More theory, just code!

- `val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b)`

//takes the list of tuples and sums the result, thus counting each word

No More theory, just code!

- `val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b)`

`//BUT WAIT WHATS THAT?`

No More theory, just code!

- val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b)

//BUT WAIT WHATS THAT?

Demo

ML packages

- Parallel distributed computing and Machine Learning complement each other well!

- you can take advantage of a large number of clusters to train models concurrently - fantastic when you have large amounts of data e.g. Google text prediction using search queries

- You can query a multiple models concurrently, or parallelise access to individual ones in order for reduced query responses for complex models

Spark MLib

- composed of common ML algorithms and utility functions for classification and regression problems, predominantly for supervised learning (but unsupervised and reinforcement learning is also supported).

- has standard implementations of ML techniques and models (including random forests, dimensionality reduction and k-means clustering etc…)

- takes advantage of distributed parallelism (using algorithms such as MapReduce) to vastly reduce the time to train models, predict and access data

Enter the Matrix

- the underlying ML type is the vector, spark supports both local and distributed vectors/matrices (with lazy evaluation for efficiency) as the interface. Underneath these are just RDDs.

- These are either dense or sparse matrices (dense ones are backed by an RDD of double values whilst sparse ones are backed by an RDD of double values and an RDD of integer indices.

- distributed matrices are made of local vectors (on local machines) which are assigned to a long index and passed via the controller from one node to another when needed. It is assumed that the local vectors are deterministic and generally small (a few columns) so that caching can occur to reduce transmission times and the space required to do so.

Useful resources(Scala)

- Spark website: http://www.scala-lang.org/index.html
- Download Scala - http://www.scala-lang.org/downloads
- hello world: http://www.scala-lang.org/old/node/166
- learnxinyminutes: http://learnxinyminutes.com/
- some of my code here: https://gist.github.com/lukeg101/8af9e97fbb76bdf1dbdd

More resources (spark)

- apache spark website: http://spark.apache.org/
- Spark RDDs API: http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.rdd.RDD
- some Art: http://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt
- DataMining Course: https://www.coursera.org/course/mmds

Resources (MLib)

- MLib: https://spark.apache.org/mllib/

- MLib types: https://spark.apache.org/docs/latest/mllib-data-types.html

- original talk: https://github.com/lukeg101/Talks/blob/master/IntroToScalaAndSparkTalk.pdf

Thanks for listening!

Any Questions?