  
第11章 大數據分析範例－書籍推薦



### 讀入資料檔與資料前處理

|  |  |
| --- | --- |
| 指令 | 說明 |
| 進入Ubuntu環境 | |
| start-all.sh | 啟動hadoop，若已經啟動，則不需要再執行此指令 |
| cd ~ | 回到家目錄 |
| wget http://www2.informatik.uni-freiburg.de/~cziegler/BX/BX-CSV-Dump.zip | 下載書籍評價資料集 |
| unzip BX-CSV-Dump.zip | 解壓縮 |
| hdfs dfs -mkdir /bookcrossing | 在HDFS建一目錄bookcrossing |
| hdfs dfs -put ~/BX-Book-Ratings.csv /bookcrossing  hdfs dfs -put ~/BX-Books.csv /bookcrossing  hdfs dfs -put ~/BX-Users.csv /bookcrossing | 將資料集上傳HDFS |
| hdfs dfs -ls /bookcrossing | 查看HDFS資料 |
| mkdir checkpoint | 建立checkpoint的目錄 |
| 進入spark-shell環境，開始處理 | |
| spark-shell | 進入spark-shell |
| sc.setCheckpointDir("checkpoint") | 設定checkpoint目錄 |
| import org.apache.spark.mllib.recommendation.{ALS, Rating, MatrixFactorizationModel} | 匯入推薦系統的程式庫 |
| val rawBookData=sc.textFile("hdfs://master:9000/bookcrossing/BX-Books.csv")  rawBookData.count  rawBookData.first | 將書籍資料讀入rawBookData |
| val bookRowRDD=rawBookData.map(line => line.replaceAll("\"","").split(";")).mapPartitionsWithIndex { (index,lines) => if (index==0) lines.drop(1) else lines }  bookRowRDD.first | 雙引號清除、使用分號分割欄位，並去除首行欄位名稱 |
| val isbnToIndex=bookRowRDD.map(x=>x(0)).zipWithIndex.collectAsMap  val bcIsbnToIndex = sc.broadcast(isbnToIndex).value  bcIsbnToIndex("0195153448") | 將ISBN書號轉成整數的索引號（ALS演算法才能處理），並建立一個對照表 |
| val indexToBkname=bookRowRDD.map(x=>(bcIsbnToIndex(x(0)), x(1))).collectAsMap  val bcIndexToBkname = sc.broadcast(indexToBkname).value  bcIndexToBkname(bcIsbnToIndex("0195153448"))  bcIndexToBkname(1) | 建立用索引查書名對照表 |

### 建立Rating資料格式

|  |  |
| --- | --- |
| 指令 | 說明 |
| val rawRatingData=sc.textFile("hdfs://master:9000/bookcrossing/BX-Book-Ratings.csv")  rawRatingData.count  rawRatingData.first | 讀入評分的原始資料 |
| val ratingRowRDD=rawRatingData.map(line => line.replaceAll("\"","").split(";")).mapPartitionsWithIndex { (index,lines) => if (index==0) lines.drop(1) else lines }  ratingRowRDD.count  ratingRowRDD.first | 基本處理 |
| val goodRatingRowRDD=ratingRowRDD.filter(x=>x(2).toInt != 0).filter(x=> bcIsbnToIndex.contains(x(1)))  goodRatingRowRDD.count | 去除評分為0和幽靈ISBN |
| val ratingsRDD=goodRatingRowRDD.map {  case Array(userid, isbn, rating) => Rating(userid.toInt, bcIsbnToIndex(isbn).toInt, rating.toDouble)  }  ratingsRDD.first | 建立Rating |

### 切割資料集

|  |  |
| --- | --- |
| 指令 | 說明 |
| val Array(bookTrainSet, bookValidSet, bookTestSet) = ratingsRDD.randomSplit(Array(0.7, 0.2, 0.1))  bookTrainSet.cache  ratingsRDD.cache  bookTrainSet.count  bookValidSet.count  bookTestSet.count | 以7:2:1的比率隨機切割為三個子資料集 |

### 模型訓練

|  |  |
| --- | --- |
| 指令 | 說明 |
| val rank=20  val numOfIter=20  val lambda=0.5  val bookModel = ALS.train(bookTrainSet, rank, numOfIter, lambda) | 設定初步的超參數組合，並模型訓練 |
| bookModel.userFeatures.take(1)  bookModel.productFeatures.take(1) | 會員特徵矩陣和書籍特徵矩陣 |
| bookModel.predict(8, 1) | 預測一下會員8對書籍1可能的評分 |

### 預測與模式評估

|  |  |
| --- | --- |
| 指令 | 說明 |
| import org.apache.spark.mllib.evaluation.RegressionMetrics  import org.apache.spark.rdd.\_ | 匯入必要的程式庫 |
| def getMSE(model: MatrixFactorizationModel, dataset: RDD[Rating])={  val userAndProd = dataset.map{ case Rating(user, product, rating)=> (user, product)}  val predict = model.predict(userAndProd).map{case Rating(user, product, rating) => ((user, product), rating)}  val realAndPredict = dataset.map{case Rating(user, product, rating) => ((user, product), rating)}.join(predict)  val realAndPredictPairs = realAndPredict.map { case ((user, product), (predict, real)) => (predict, real)}  val regMetrics = new RegressionMetrics(realAndPredictPairs)  val MSE=regMetrics.meanSquaredError  MSE  } | 計算MSE的副程式，輸入一個已經訓練好的模型和資料集（通常是驗證資料集） |
| getMSE(bookModel, bookTrainSet)  getMSE(bookModel, bookValidSet) | 分別用bookTrainSet和bookValidSet計算 MSE |

### 參數最佳組合與最佳模型

|  |  |
| --- | --- |
| 指令 | 說明 |
| def modelAndMSE(rank: Int, numOfIter:Int, lambda:Double)={  val bookModel = ALS.train(bookTrainSet, rank, numOfIter, lambda)  val MSE=getMSE(bookModel, bookValidSet)  println(f"MSE:$MSE%2.4f <== Rank:$rank%2d, Inter:$numOfIter%2d, 　Lambda:$lambda%2.2f")  (MSE, bookModel)  } | 訓練某一參數組合（rank、numOfIter、lambda）的模型，並計算MSE，將結果MSE和bookModel返回 |
| var bestMSE=Double.MaxValue  var bestModel:MatrixFactorizationModel=null  for(rank<-Array(20, 30); numOfIter<-Array(20, 30); lambda<-Array(0.5, 　1)) {  val (tempMSE, tempModel)=modelAndMSE(rank, numOfIter, lambda)  if(tempMSE < bestMSE) {  bestMSE=tempMSE  bestModel=tempModel  }  } | 求取各超參數組合的MSE |
| bestModel.predict(8, 1) | 用最佳模式預測一下會員8對書籍1可能的評分 |
| getMSE(bestModel, bookTestSet) | 用最佳模式和測試資料集 bookTestSet 來計算 MSE |

### 使用最佳模型進行推薦

|  |  |
| --- | --- |
| 指令 | 說明 |
| val recomProd=bestModel.recommendProducts(8, 10)  recomProd.foreach(x=>println(f" 評分 : ${x.rating}%2.4f, 書名 : ${bcIndexToBkname(x.product)}%s")) | 為會員8推薦10本他可能有興趣的書 |
| val recomUser=bestModel.recommendUsers(1, 10)  recomUser.foreach(x=>println(f" 評分 : ${x.rating}%2.4f, 使用者編號 : ${x.user}%s")) | 著為書籍1推薦可能有興趣的10個會員 |