大数据编程模型和使用技巧

机器学习与图计算

陈一帅

yschen@bjtu.edu.cn

北京交通大学电子信息工程学院

内容

- 机器学习
- 图计算

大数据机器学习

- 机器学习已成为云计算应用的核心
- 机器学习最近取得的重大突破来自几个方面
  - 大数据
  - 算法进步
  - 更快计算平台（GPU）

四个基本概念：DataFrame

- 以有效的方式保存矢量和其他结构化数据类型
- 与 Pandas DataFrames 类似，共享一些操作
- 它们是分布式对象，是执行图的一部分
- 可将它们转换为 Pandas DataFrame，就可以用 Python 访问它们

四个基本概念：Transformers

- 将一个 DataFrame 转换为另一个 DataFrame 的运算符
- 它们是执行图上的节点，因此在执行整个图之前不会对其进行评估
- 如
  - 将文本文档转换为向量
  - 将 DataFrame 的列从一种形式转换为另一种形式
  - 将 DataFrame 拆分为子集

四个基本概念： Estimators

- 封装 ML 和其他算法
- fit()方法将 DataFrame 和参数传递给学习算法以创建模型

Pipeline

- 通常是线性的，但也可以是有向无环图
- 链接 Transformers 和 Estimators，指定一个 ML 工作流
- 用 fit() 训练完估算器后，Pipeline 就是一个模型，具有 transform() 方法，可对新案例进行预测

Spark MLib

- 步骤 1
  - 输入数据分为两个子集：训练数据与测试数据
  - 在进入计算或学习引擎之前，两者都存储在数据存储器中
- 步骤 2
  - 数据预处理，例如过滤，挖掘，数据聚合，特征提取，模式识别以及某些转换操作
- 步骤 3
  - 使用云计算和存储资源的学习引擎
  - 包括数据清理，模型训练以及在监督下向模型开发的转变。
- 步骤 4
  - 学习模型的构建，适应环境满足预测或分类等学习目标的环境问题
- 步骤 5
  - 通过制定决策或预测进行的训练和测试阶段

交叉验证

- 机器学习中的一项重要任务是模型选择，或使用数据为给定任务找到最佳模型或参数。这也称为 Tunning
  - Pipeline 可以轻松地一次调整整个 Pipeline，不必分别调整其中的每个元素，简化了模型选择
  - MLlib 支持使用 CrossValidator 类进行模型选择，该类具有一个估计器，一组 ParamMap 和一个评估器

交叉验证

- CrossValidator 首先将数据集划分为一组 folds，它们将被用作单独的训练和测试数据集
  - 如 k = 3 folds，就会生成 3 对（训练，测试）数据集对，每对使用三分之二的数据用于训练，另外三分之一的数据用于测试。
- CrossValidator 遍历 ParamMaps 集。对于每个 ParamMap，它训练给定的估算器并对其进行评估，选择产生最佳评估指标的 ParamMap 作为最佳模型
- 最后，CrossValidator 使用最佳的 ParamMap 和整个数据集来训练最终的估算器

示例

- 创建 DataFrame，包含由矢量表示的标签和多个特征

```
df = sqlContext.createDataFrame
    (data, [“label”, “features”])
```

- 设置算法参数。在这里，我们将 LR 的迭代次数设为 10

```
lr = LogisticRegression(maxIter = 10)
```

示例

- 从数据中训练模型

```
model = lr.fit(df)
```

- 将数据集送入训练好的模型，预测每个点的标签，显示结果

```
model.transform(df).show()
```

内容

- 机器学习
- 图计算

Spark GraphX

- Spark Core 支持的分布式图计算框架
- 提供了表达图形计算的 API，可对 Pregel 抽象进行建模
- 为这种抽象提供了优化的运行时支持。”
- 将提取，转换，加载（ETL）函数，探索性分析和迭代图计算统一
- 能够对 RDD 进行有效的转换和图连接
- 用户可以使用 Pregel API 编写自定义的迭代图算法

GraphX

- 将图计算嵌入分布式数据流框架中
- 将图计算提炼到特定的 join-map-groupBy 数据流模式
- 通过将图计算减少到特定的模式，用户可以确定系统优化的关键路径

提供的图算法

- PageRank
- 连通图
- 标签传播
- SVD++
- 强连接组件
- 三角形计数

例：计算用户 PageRank

- Spark GraphX PageRank 的社交网络数据集示例
  - 在 graphx/data/users.txt 中提供了一组用户
  - 在 graphx/data/followers.txt 中提供了一组用户之间的关系
- 将边缘作为图形加载

```
val graph = GraphLoader
  .edgeListFile(sc, “graphx/data/followers.txt”)
```

例：计算用户 PageRank

- 运行 PageRank

```
val ranks = graph.pageRank(0.0001).vertices
```

- Join 用户名和 Rank

```
val users = sc.textFile(“graphx/data/users.txt”)
  .map {line => val fields
    = line.split(“,”) (fields(0).toLong, fields(1))}
val ranksByUsername = users.join(ranks)
  .map {case (id, (username, rank)) => (username, rank)}
```

- 打印结果

```
println(ranksByUsername.collect().mkString(“\n”)),
```

mrTriplet

- 图并行计算 mrTriplet 的过程
  - mrTriplets 运算符是三元组视图上 map 和 groupBy 数据流运算符的合成
  - 计算每个顶点用户的较老关注者的数量
  - 用户定义的 map 函数会应用于每个三元组，生成一个值
  - 然后使用用户定义的二元聚合函数在目标顶点将其聚合

```
val graph: Graph[User, Double]
def mapUDF(t: Triplet[User, Double]) =
    if (t.src.age > t.dst.age) 1 else 0
def reduceUDF(a: Int, b: Int): Int = a + b
val seniors: Collection[(Id, Int)] =
    graph.mrTriplets(MapUDF, reduceUDF)
```

小结

- 机器学习
- 图计算

练习

- Spark 机器学习
- Spark 图计算