**Advanced Analytics with Spark**

0. 一些点

* 广播变量（broadcast）
  + Spark执行一个stage时，会为待执行函数建立闭包，也就是阶段所有任务所需信息的二进制形式。这个闭包包括驱动程序里函数饮用的所有数据结构。Spark把这个闭包发送到集群的每个executor上。
  + 当许多任务需要访问同一个不可变的数据结构时，应该使用广播变量：
    - 在每个executor上将数据缓存为原始的Java对象，不用为每个任务执行反序列化
    - 在多个作业和阶段之间缓存数据

3.音乐推荐 【协同过滤】【ALS(Alternating Least Square)】

* 协同过滤是一种根据用户对各种产品的交互与评分来推荐新产品的推荐系统技术
* 本例用到潜在因素模型中的矩阵分解模型。
  + 数学上，这些算法把用户和产品数据当成一个大矩阵A，矩阵第i行和第j列上的元素有值，代表用户i播放过艺术家j的音乐。A是稀疏矩阵，矩阵分解算法把A分解为两个小矩阵X和Y的乘积，X和Y都非常瘦。因此，矩阵分解算法只能是某种近似。
  + A=XYT 通常没有解，所以XYT应该尽可能逼近A，但他不能也不应该完全复制A。
  + 求解X和Y时，使用交替最小二乘算法（ALS）
    - 主要思想：把Y初始化为随机行向量矩阵。接着运用简单的线性代数，就能在给定A和Y的条件下求出X的最优解。针对每一行，我们希望最小化。同理，再由得到的X来计算每行的Y。就这样交替往复，最终X和Y会收敛得到一个合适的结果。
    - 算法在以下三点时速度非常快：
      * 稀疏的输入数据
      * 可以用简单的线性代数运算求最优解
      * 数据本身可并行化
    - ALS.trainImplicit()超参数
      * rank，模型的潜在因素个数
      * iterations，矩阵分解迭代的次数
      * lambda，标准的过拟合参数
      * alpha，控制矩阵分解时，被观察到的“用户-产品”交互相对没被观察到的交互的权重

4.用决策树算法预测森林植被【Decision Tree】

* Covtype数据集：t.cn/R2wnIsI，一个csv格式的covtype.data.gz
* 记录了科罗拉多州不同地块的森林植被类型
  + 海拔、坡度、到水源的距离、遮阳情况、土壤类型
  + 预测森林植被类型
* Spark MLlib将特征向量抽象为LabeledPoint，由一个包含多个特征值的Spark MLlib vector和一个称为label的目标值组成。目标位double类型，vector本质上是对多个double类型值的抽象。因此，LabeledPoint只适用于数值型特征，但是经过适当编码也可以用于类别型特征(1 of k)。
* Precision rate & recall rate
  + 50个样本中20个为正，分类器将50个中的10个标为正，这十个中间只有4个真的为正。
  + Precision: 4/10=0.4
  + Recall: 4/20=0.2
* 决策树的超参数
  + 最大深度
    - 分类器为了对样本进行分类所作的一连串判断的最大次数
    - 限制最大深度可以防止过拟合
  + 最大桶数
    - 决策总是采用相同形式
      * 数值型特征（“特征值>= ”的形式）
      * 类别型特征（“特征在（值1，值2，…） ”的形式）
  + 不纯性度量
    - 好规则把训练集数据的目标值氛围相对是同类或“纯”的子集。
    - 选择最好规则也就意味着最小化规则对应的两个子集的不纯性
    - 常见度量
      * Gini不纯度
        + 如果子数据集中所有样本都属于一个类别，Gini不纯度的值为0
        + 每个类别的样本数相同时Gini不纯度最大
      * 熵
        + 来源于信息论，代表了子集中目标取值集合的不确定程度。
* Decision Tree代码中的参数Map[Int,Int]
  + 指明了输入数据中每个类别型特征预期的不同取值的个数
  + 参数取为空Map()，表示算法不需要把任何特征作为类别型，也就是说所有特征都是数值型的。
* 决策树使用一些启发式策略，能够聪明地找到需要实际考虑的少数规则。在选择规则的过程中也涉及一些随机性；每次只考虑随机选择少数特征，并且只考虑训练数据中一个随机子集。牺牲准确度的情况下换回了速度的大幅提升，但也意味着每次决策树算法构造的树都不相同。
* 最好的树不只有一棵，应该有很多棵，每一棵都能对正确目标值给出合理、独立且互不相同的估计。这些树的集体平均预测应该比任一个个体预测更接近正确答案。这就是**决策森林**的关键所在。
  + val forest=RandomForest.trainClassifier(trainData,7,Map(10->4,11->40),20, “auto”, “entropy”,30,300)
  + 多出的两个新参数
    - 1.构建多少棵树
    - 2.特征决策树每层的评估特征选择策略（“auto”），不考虑全部特征，只考虑一个子集。

5.基于K均值聚类的网络流量异常检测【KMeans】

* 异常
  + 本质上异常属于“未知的未知”
  + 要应用到非监督学习
* K均值聚类 K means clustering
  + 将一系列样本分割成K个不同的类簇(K是模型的输入参数)
  + 类簇的中心称为质心，是类簇中所有点的算术平均值，因此算法取名为K均值
  + 开始时选择一些数据点作为类簇的质心，然后把每个数据点分配给最近的质心，接着对每个类簇计算改类簇所有数据点的平均值，并将其作为该类簇的新质心。
* KMeans Model
  + Predict 返回数据所属的簇
  + clusterCenters 返回质心
  + setRuns()
  + setEpsilon()

6.基于潜在语义分析算法分析维基百科

* 潜在语义分析（Latent Semantic Analysis），一种自然语言处理和信息检索技术，目的是更好地理解文档语料库以及文档中词项的关系。把语料库提炼成一组相关概念，每个概念捕捉了数据中一个不同的主题，每个概念由三个属性组成：语料库中文档的相关度、语料库中词项的相关度、概念对描述主题的重要性评分。
* LSA在降维过程中使用一种称为奇异值分解（SVD, Singular Value Decomposition）的线性代数技术
  + 首先根据词项在每个文档中的出现次数构造一个词项-文档矩阵。矩阵中每个文档对应一列，每个词项对应一行，矩阵的每个元素代表某个词项在对应文档中的重要性。
  + SVD将矩阵分解成三个矩阵，其中一个矩阵代表文档中出现的概念，另一个代表词项对应的概念，还有一个代表每个概念的重要度。这三个矩阵的结构可以让我们通过去掉最不重要的概念所对应的行和列而获得原始剧真的一个低阶近似。将这些低阶矩阵相乘就可以得到原始矩阵的近似，去掉的概念越多，矩阵越失真。
* 词项-文档矩阵
  + TF-IDF（term frequency times inverse document frequency）表示矩阵中的权重
    - def termDocWeight(termFrequencyInDoc: Int, totalTermsInDoc: Int, termFreqInCorpus: Int, totalDocs: Int): Double ={

val tf=termFrequencyInDoc.toDouble/totalTermsInDoc

val docfreq=totalDocs.toDouble/termFreqInCorpus

val idf=math.log(docFreq)

tf\*idf

}

* + - 一个词项在文档中出现的次数越多，它对于文档的重要性越高
    - 词项是不平等的，文档中出现语料库中罕见词项的意义比出现常见词项更大，因此指标就是词项在所有语料库重出现次数的倒数。
    - 词项在语料库中的频率分布往往呈指数型，算法对逆文档频率取对数。文档频率的差别就从乘数级变成了加数级。
  + 词形归并
    - the和is之类的常用词不会为模型提供有用信息，却占用大量空间。去掉这些stop word，可以节省空间，并提高忠实度
    - 相同意思的词项可能有不同词形（名词和它的形容词，一个词的单复数），把这些不同曲折词缀合并成单个词项，即词干还原（stemming）或词形归并（lemmatization）。