数据挖掘基础算法

数据挖掘并行算法研究的重要性

海量数据

大数据隐含着更准确的事实

基于 MapReduce 的 K-Means 聚类算法

K-Means 聚类算法

聚类：将给定的多个对象分为若干组，组内的对象是相似的，组间的对象是不相似的，划分后的组称为簇（cluster）

数据点可以有欧氏空间的和非欧的，簇的表示方法可以是取各点的平均值（欧氏）或是代表点（非欧）

对于相似度的计算，欧氏空间可以直接计算欧氏距离，非欧空间有

- Jaccard 距离：$1-|S\cap T|/|S\cup T|$
- Cosine 距离：向量夹角大小
- Edit 距离：字符串的编辑距离

基于划分的聚类方法：给定 N 个对象，构造 K 个分组，满足：每个分组至少一个对象，每个对象属于且仅属于一个分组

K-Means 算法可以描述为

- 输入 N 个对象和期望生成的簇的个数 K
- 选取 K 个点作为初始的 cluster center
- 对每个点 p
    - 计算 p 到各个 cluster center 的距离
    - 将其归入最近的 cluster
- 重新计算 cluster center
- 重复直至满足结束条件

其局限性在于初始 cluster center 的选择会影响最终的聚类结果

MapReduce 设计思路

关键在于：处理每个点时，只需要知道 cluster center，不需要知道其余点的信息

Map：读取 cluster center，判断每个点属于哪一个 cluster

Reduce：计算新的 cluster center

只需要共享迭代计数和 cluster center 信息

具体算法为

- Map：对于点 p，选择其属于的 cluster，输出 `<ClusterID, (p, 1)>`
- Combiner：对于相同 cluster 的点，计算其均值 pm 及其个数 n，输出 `<ClusterID,(pm, n)>`
- Reduce：根据输入计算并输出 `<ClusterID, pm>` 作为新的 cluster center

可选的终止条件有

- 达到迭代次数
- 每个点所属 cluster 不变
- 根据具体应用设置

基于 MapReduce 的分类并行算法

分类问题

从一组带有分类标记的训练样本数据集来预测一个测试样本的分类结果

每个训练样本是一个三元组：标识符，特征值，分类标记

KNN 分类并行算法

KNN 基本思想：计算测试样本到各训练样本的距离，取距离最近的 K 个，根据这 K 个样本的分类进行投票

可以根据距离进行加权投票

MapReduce 设计思路：测试样本分块处理，训练样本放在 Distributed Cache 共享

Map：计算测试样本最近的 K 个训练样本，投票得出类别

Reduce：直接输出

朴素贝叶斯分类并行算法

如果每个样本用一个 n 维向量描述，如果有 m 个类，则对于样本 X 如果被分到 Y_i 类，则一定有

$$P(Y_i|X) > P(Y_j|X), j \neq i$$

根据贝叶斯定理

$$P(Y_i|X) = \frac{P(X|Y_i)P(Y_i)}{P(X)}$$

且 $P(X)$ 为常数，即

$$P(Y_i|X) = P(X|Y_i)P(Y_i)$$

其中如果认为 X 的特征值分量彼此独立，则有

$$P(X|Y_i) = \prod P(x_i|Y_i)$$

可以根据训练样本求得

则核心计算为计算训练集中 Y_i 的频数和 x_j 在 Y_i 中的频数

在 MapReduce 中就是简单的频数统计，由 Map 发射，Reduce 汇总统计

SVM 短文本多分类并行算法

对大量短文本分类，每个文本是一个高维的特征向量

思路：对每个类生成一个 binary 的分类器（是否属于该类），用多个分类器对测试样本分类并打分，分类为是且打分最高的是预测结果，如果打分低于阈值则不属于已知类别

MapReduce 的频繁项集挖掘算法

频繁项集问题

对于一个订单数据库，设 N 为所有订单数，M 是所有包含指定项集 I 中所有元素的订单数，则定义

$$sup(I) = \frac{M}{N}$$

如果 $sup(I)$ 不小于一个指定阈值，则认为其是一个频繁项，频繁项集挖掘即是寻找所有频繁项

现有算法

Apriori 算法：每轮迭代寻找频繁 k-项集（第一轮是 1-项集），根据前一轮的频繁项集生成下一轮的候选者

SON 算法：将数据库划分成多个不相交的部分，分别计算各部分的频繁项集，然后 merge，基于两个引理

- 不是局部频繁则一定不是全局频繁
- 全局频繁一定是局部频繁

PSON：基于 MapReduce 的并行算法

基本思路是并行的 SON 算法

Map 寻找局部的频繁项集

Reduce 合并处理相同长度的局部频繁项集，生成全局频繁项集

分为两个 job 处理

- Job1：生成全局频繁项集的候选
  - Map：生成局部频繁项集（使用 Apriori 算法），然后发射
  - Reduce：合并所有相同的频繁项集输入，只输出一个结果
- Job2：生成全局频繁项集
  - Map：根据 Job1 的结果，统计每个候选在本地的划分出现的频次
  - Reduce：统计每个项集的频次，不小于指定阈值的是频繁项集