MapReduce数据挖掘基础算法(I)

K-Means聚类算法

摘要

- 为什么选择数据挖掘作为并行计算的研究点
- K-Means聚类算法介绍
- K-Means算法为什么适合使用并行方法
- 基于MapReduce的K-Means并行算法

数据挖掘

- 定义：数据挖掘是通过对大规模观测数据集的分析，寻找确信的关系，并将数据以一种可理解的、且利于使用的新颖方式概括数据的方法。
- 核心目标：发现知识，侧重于解释性和实用性
- 数据挖掘的特征之一：海量数据
  - Small data does not require data mining, large data causes problems.
- 研究发现大数据隐含着更为准确的事实。
- 可见，数据挖掘是并行计算中值得研究的一个领域。

数据挖掘

- 研究发现大数据隐含着更为准确的事实

2001年微软研究院的Banko and Brili等研究发现数据越大，机器学习的精度越高；当数据不断增长时，不同算法的分类精度趋向于相同！

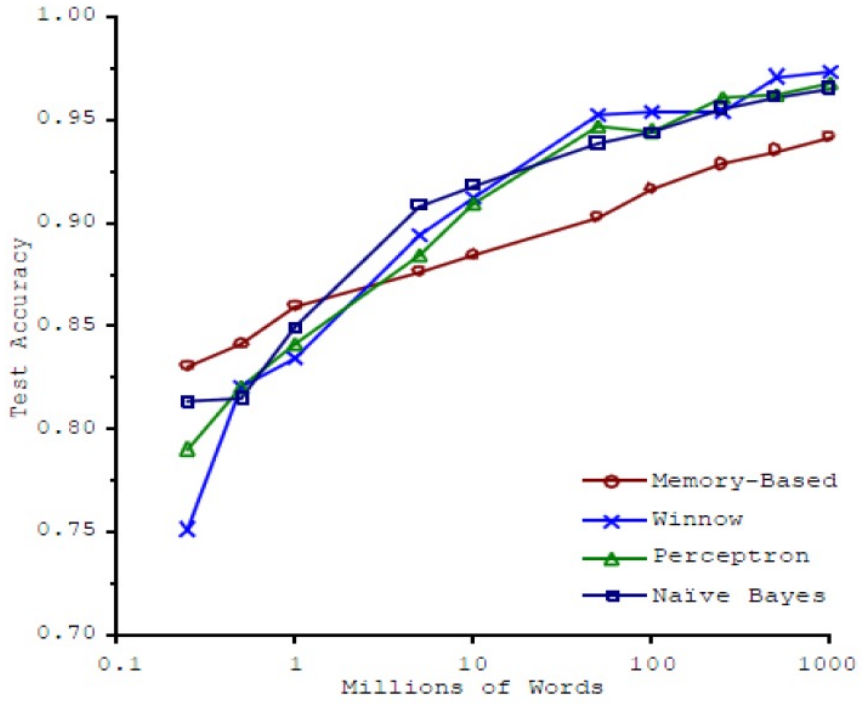


Figure 1. Learning Curves for Confusion Set Disambiguation

M. Banko and E. Brili (2001). **Scaling to very very large corpora for natural language disambiguation.** *ACL 2001.*

数据挖掘

- 研究发现大数据隐含着更为准确的事实

2007年Google公司的Brants等基于MapReduce研究了一个2万亿单词训练数据集的语言模型，发现大数据集上的简单算法能比小数据集上的复杂算法产生更好的结果！

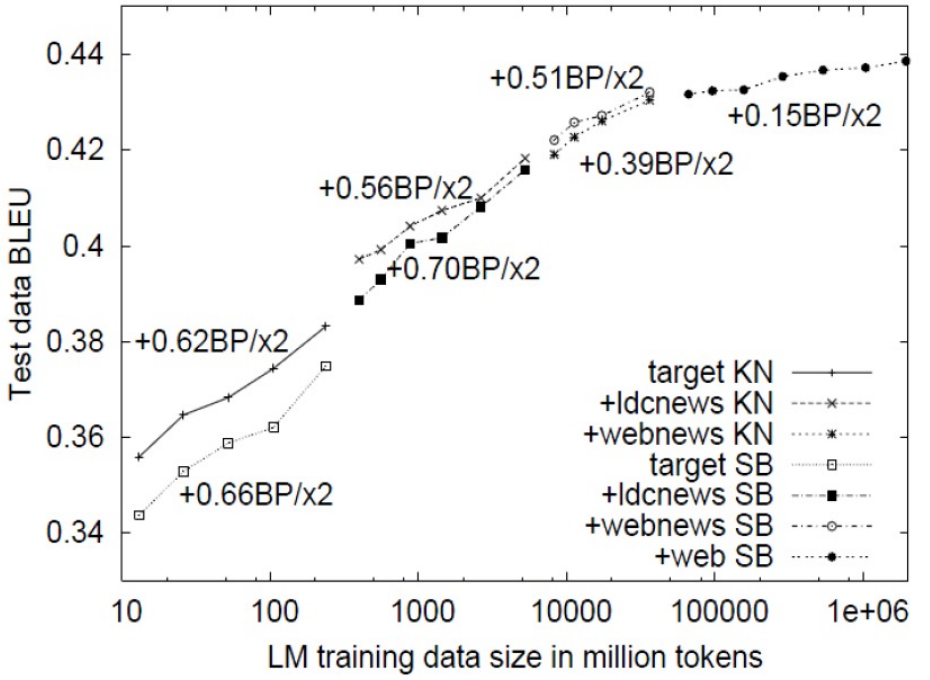


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

T. Brants, A.C. Popat, et al. (2007). **Large language models in machine translation.** *In EMNLP-CoNll 2007.*

机器学习

- 定义：通过算法让计算机从数据中学习规律，并基于学习结果做出预测或决策。
- 核心目标：构建泛化能力强的模型，侧重于预测精度和自动化。
- 机器学习是数据挖掘的核心工具之一
  - 数据挖掘的许多任务（如分类、回归、聚类）依赖机器学习算法（如决策树、SVM、神经网络）。

数据挖掘 vs. 机器学习

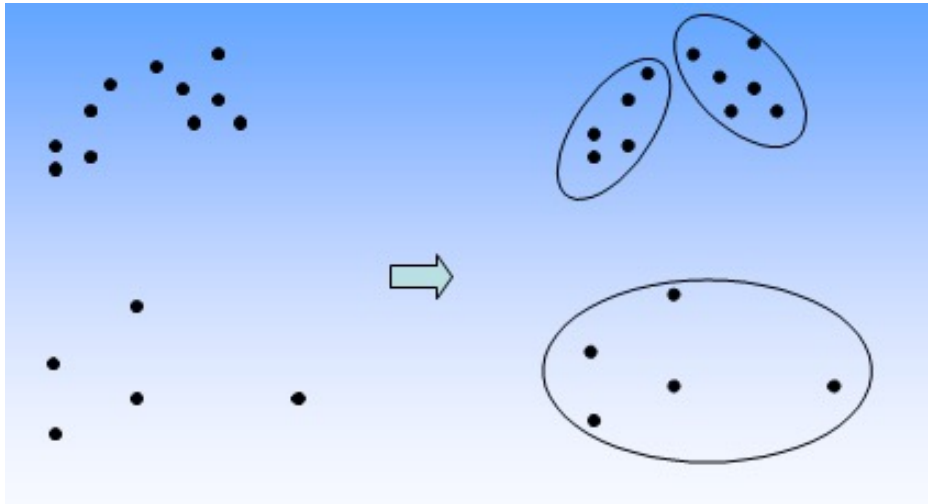
- 数据挖掘不限于机器学习，数据挖掘还包含非机器学习技术，如：
  - 统计分析（假设检验、相关性分析）
  - 数据库技术（OLAP、SQL查询）
  - 可视化（数据分布探索）
  - 传统规则挖掘（Apriori算法找关联规则）
- 大数据领域的研究为数据挖掘提供数据管理技术（如数据库等），而机器学习和统计学的研究为数据挖掘提供数据分析技术。

聚类的应用领域

- 市场营销
  - 给定一个很大的顾客交易集，找出有类似购买行为的顾客分组
- 文档分类
  - 对Web日志数据聚类，发现有类似访问模式的分组
- 保险
  - 通过识别可能的欺诈行为找出平均索赔支出很高的车险投保人群
- …

聚类过程

- 定义：将给定的多个对象分成若干组，组内的各个对象是相似的，组间的对象是不相似的。进行划分的过程就是聚类过程，划分后的组称为簇（cluster）。
- 几种聚类方法：
  - 基于划分的方法；
  - 基于层次的方法；
  - 基于密度的方法；
  - ……



点、空间和距离

- 点集是一种适合于聚类的数据集，每个点都是某空间下的对象。
  - 欧式空间下的点就是实数向量；
  - 向量的长度是空间的维数；
  - 向量的分量通常称为所表示点的坐标。
- 能够进行聚类的所有空间下都有一个距离测度，即给出空间下任意两点的距离。
  - 距离永远非负，只有点到自身的距离为0；
  - 距离具有对称性；
  - 距离遵守三角不等式。

数值类型

- 数据点的类型可分为：
  - 欧氏（Euclidean）空间：空间中的点的平均总是存在，并且也是空间中的一个点。一般用欧几里得距离来衡量两个点之间的距离。
  - 非欧空间：Jaccard距离，Cosine距离，Edit编辑距离等多种距离衡量方法。

- 这二者在数据的表示以及处理上有较大的不同：
  - 怎样来表示cluster?
  - 怎样来计算相似度?

Cluster的表示

- 欧氏空间：
  - 取各个数据点的平均值（centroid）
- 非欧空间：
  - 取某个处于最中间的点
  - 取若干个最具代表性的点（clustroid）
  - ……

相似度（距离）的计算

- 欧氏空间：可以有较为简单的方法

$$d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- 非欧氏空间：通常不能直接进行简单的数字计算
  - Jaccard 距离：两个集合中不同元素占所有元素的比例

$$J_\delta = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

  - Cosine距离：两个向量的夹角大小

$$\cos A = \frac{<b,c>}{|b||c|}, \quad sim(X,Y) = cos\theta = \frac{\vec{x} \cdot \vec{y}}{\|x\| \cdot \|y\|}$$

  - Edit 距离：适合于string类型的数据
  - Hamming距离：两个向量中不同分量的个数

基于划分（partitioning）的聚类方法

- 给定N个对象，构造K个分组，每个分组就代表一个聚类。

- 这K个分组满足以下条件：
  - 每个分组至少包含一个对象；
  - 每个对象属于且仅属于一个分组。

- K-Means算法是最常见和典型的基于划分的聚类方法

注：本节课只讨论欧氏空间里的K-Means聚类方法

K-Means算法

输入： 待聚类的N个数据点，期望生成的聚类的个数K

输出： K个聚类

----算法描述----------

选出K个点作为初始的cluster center

Loop:

对输入中的每一个点p: {

计算p到各个cluster center的距离;

将p归入最近的cluster;

}

重新计算各个cluster的中心

如果不满足停止条件，goto Loop；否则，停止

过程示例（1）

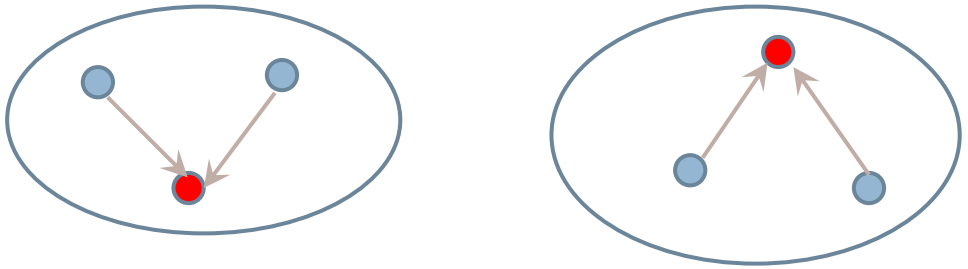
初始数据
K = 2

选择初始中心

第1次聚类：计算距离

过程示例（2）

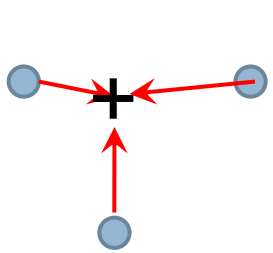
第1次聚类：归类各点

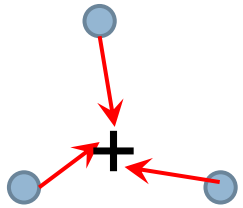


重新计算聚类中心

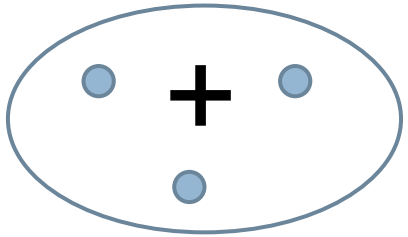
过程示例（3）

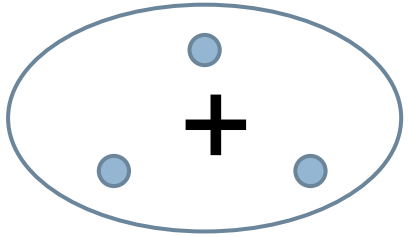
第2次聚类：计算距离





第2次聚类：归类各点





聚类无变化，迭代终止

K-Means是个不断迭代的过程

- 第i轮迭代：
  - 生成新的clusters，并计算cluster centers
- 第i+1轮迭代：
  - 根据第i轮迭代中生成的clusters和计算出的cluster centers，进行新一轮的聚类

◈ 如此不断迭代直到满足终止条件

K-Means算法的局限性

- 对初始cluster centers的选取会影响到最终的聚类结果
- 由此带来的结果是：能得到局部最优解，不保证得到全局最优解
- 相似度计算和比较时的计算量较大

K-Means计算性能的瓶颈

- 如果样本数据有n个，预期生成k个cluster，则K-Means算法t次迭代过程的时间复杂度为O(n*k*t)，需要计算n*t*k次相似度
- 如果能够将各个点到cluster center相似度的计算工作分摊到不同的机器上并行地计算，则能够减少计算时间
- 利用MapReduce将K-Means聚类过程并行化

考虑数据相关度

- 在进行K-Means聚类中，在处理每一个数据点时
- 只需要知道：各个cluster 的中心信息
- 不需要知道：关于其他数据点的任何信息
  - 所以，如果涉及到全局信息，只需要知道关于各个cluster center的信息即可

并行化改造的出发点

- 将所有的数据分布到不同的MapReduce节点 上，每个节点只对自己的数据进行计算
- 每个Map节点能够读取上一次迭代生成的cluster centers，并判断自己的各个数据点应该属于哪一个cluster
- Reduce节点综合每个节点计算出的相关数据，计算出最终的实际cluster centers

需要全局共享的数据

- 每一个节点需要访问如下的全局文件
  - 当前的迭代计数
  - K个 如下结构

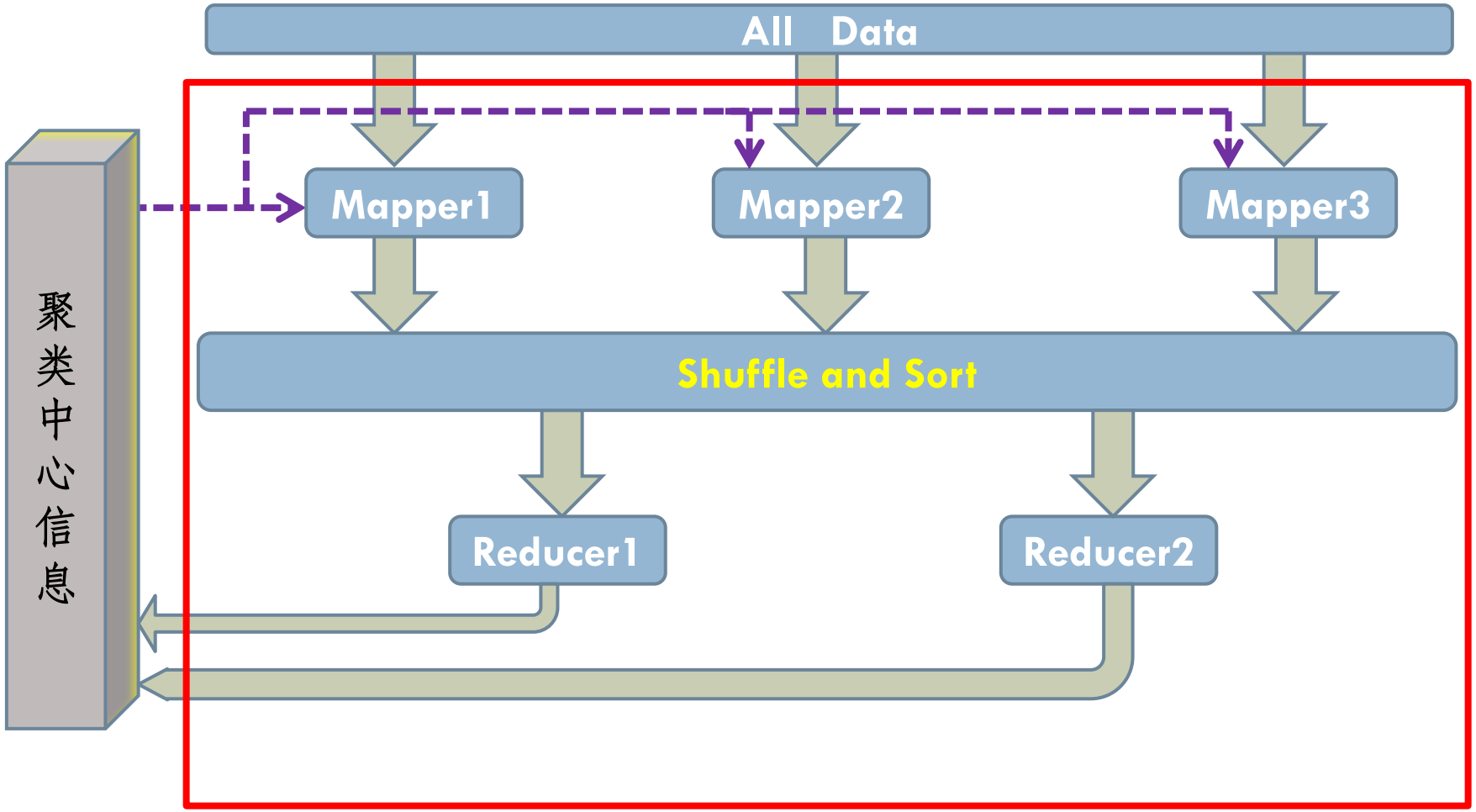
    cluster id

    cluster center

    属于该cluster center的数据点的个数

- 这是唯一的全局文件

架构图


All Data
Mapper1
Mapper2
Mapper3
Shuffle and Sort
Reducer1
Reducer2
聚类中心信息

Map阶段的处理

- Mapper伪代码

```
class Mapper
  setup(…)
  {
     读出全局的聚类中心数据 → Centers
  }

  map(key, p)  // p为一个数据点
   {
    minDis = Double.MAX VALUE;
    index = -1;
    for i=0 to Centers.length
    {
       dis= ComputeDist(p, Centers[i]);
       if dis < minDis
       {   minDis = dis;
           index = i;
       }
    }
    emit(Centers[i].ClusterID, (p,1));
  }
```

思考题：

在map中，最后一行emit(Centers[i].ClusterID, (p,1))语句中，为什么输出值必须是(p,1)而不能是p？如果写成p会带来什么问题？

Map阶段的处理

- Combiner伪代码

```
class Combiner
  reduce(ClusterID, [(p1,1), (p2,1), ...])
  {
     pm = 0.0;
     n = 数据点列表[(p1,1), (p2,1), ...]中数据点的总个数;
     for i=0 to n
        pm += p[i];
     pm = pm / n;  // 求得这些数据点的平均值
     emit(ClusterID, (pm, n));
  }
```

Reduce阶段的处理

- Reducer伪代码

```
class Reducer
  reduce(ClusterID, value = [(pm1,n1),(pm2,n2) ...])
  {
    pm = 0.0;   n=0;
    k = 数据点列表中数据项的总个数;
    for i=0 to k
    {    pm += pm[i]*n[i];  n+= n[i]; }
    pm = pm / n;  // 求得所有属于ClusterID的数据点的均值
    emit(ClusterID, (pm,n)); // 输出新的聚类中心的数据值
  }
```

用不用Combiner仅仅会影响性能，不能改变计算结果。因此，Combiner输出时不允许改变Map输出键值对中Value的格式和类型，否则会出错

数据举例说明

Object	Attribute 1 (x)	Attribute 2(y)
A	1	1
B	2	1
C	4	3
D	5	4

- 令 k = 2，欲生成 cluster-0 和 cluster-1
- 随机选取A(1,1)作为cluster-0的中心，C(4,3)作为cluster-1的中心
- 假定将所有数据分布到2个节点 node-0 和 node-1 上，即

  node-0 ： A(1,1)和C(4,3)

  node-1 ： B(2,1)和D(5,4)

准备开始K-Means聚类

- 在开始之前，首先创建一个如前所述的全局文件

Iteration No	0	
Cluster id	cluster 中心	# of data points assigned
cluster -0	A(1, 1)	0
cluster-1	C(4, 3)	0

Map阶段

- 每个节点读取全局文件，以获得上一轮迭代生成的cluster centers等信息
- 计算本节点上的每一个点到各个cluster center的距离，选择距离最近的cluster
- 为每一个数据点，发射键值对<CluterID, (数据点自身, 1)>

Map： Iteration 1

- 计算各个数据点到各个cluster的距离，假定是如下的结果

	数据点	到各个 cluster 的距离比较		Assigned to
		cluster-0	cluster-1	
Node-0	A(1, 1)	近		cluster-0
Node-0	C(4,3)		近	cluster-1
Node-1	B(2,1)	近		cluster-0
Node-1	D(5,4)		近	cluster-1

node-0输出：

<cluster-0, [A(1,1),1]>

<cluster-1, [C(4,3),1]>

node-1输出：

<cluster-0, [B(2,1),1]>

<cluster-1, [D(5,4),1]>

Combine阶段

- 利用combiner减少map阶段产生的大量的ClusterID相同的键值对<ClusterID, (p,k)>
- Combiner计算要emit的所有数据点的均值，以及这些数据点的个数
- 然后，为每一个cluster发射新的键值对<ClusterID, (pm, n)>

Combine： Iteration 1

- Map的输出（即Combiner的输入）：

node-0 发射：
<cluster-0, [A(1,1),1]>
<cluster-1, [C(4,3),1]>

node-1发射：
<cluster-0, [B(2,1),1]>
<cluster-1, [D(5,4),1]>

- Combiner的输出
  - <ClusterID, (pm, n)>

node-0发射：
<cluster-0, [(1,1),1]>
<cluster-1, [(4,3)],1]>

node-1发射：
<cluster-0, [(2,1),1]>
<cluster-1, [(5,4),1]>

Reduce阶段

- 由于map阶段emit的key是ClusterID，所以每个 cluster的全部数据将被发送同一个reducer，包括：
  - 该cluster 的ID
  - 该cluster的数据点的均值，及对应于该均值的数据点的个数

- 然后经计算后输出
  - 当前的迭代计数
  - ClusterID
  - 新的Cluster Center
  - 属于该Cluster Center的数据点的个数

Reduce： Iteration 1

- 在上一阶段，Combiner的输出

node-0 发射：
<cluster-0, [(1,1),1]>
<cluster-1, [(4,3),1]>

node-1发射：
<cluster-0, [(2,1),1]>
<cluster-1, [(5,4),1]>

- 两个reducer分别收到

Reducer-0：
<cluster-0, [(1,1),1]>
<cluster-0, [(2,1),1]>

Reducer-1：
<cluster-1, [(4,3),1]>
<cluster-1, [(5,4),1]>

Reduce： Iteration 1

- 计算可得

- cluster-0的中心 = ((1*1+2*1)/(1+1), (1*1+1*1)/(1+1))
  = (1.5,1)

- cluster-1的中心 = ((4*1+5*1)/(1+1), (3*1+4*1)/(1+1))
  = (4.5,3.5)

Reduce： Iteration 1

- Reducer输出

Iteration No	1	
Cluster id	cluster 中心	# of data points assigned
cluster -0	(1.5,1)	2
cluster-1	(4.5,3.5)	2

第1轮迭代结束

- 下面开始第2轮迭代，此时，全局文件已经更新为：

Iteration No	1	
Cluster id	cluster 中心	# of data points assigned
cluster -0	(1.5,1)	2
cluster-1	(4.5,3.5)	2

终止迭代

- 在第i次迭代后，已经生成了K个聚类。如果满足了终止条件，即可停止迭代，输出K个聚类
- 终止条件：
  - 设定迭代次数；
  - 均方差的变化（非充分条件）
  - 每个点固定地属于某个聚类
  - 其他设定条件……
- 与具体的应用高度相关

算法设计实现小结

- 利用MapReduce来并行化K-Means聚类过程是可行的
- 每个节点计算一部分数据的归属，从而实现并行
- 数据间是无关的，但是数据和聚类中心是相关的，因此需要全局文件，但不构成性能瓶颈
- 没有因为并行而降低了算法的精确度（每一个点均保证与每一个cluster center进行了比较）

Open Problem

NetFlix

百万美元

大奖赛

Netflix公司

- 美国的一家电影在线租赁公司
- 拥有大量用户的影评记录
- 影片推荐系统基于这些影评记录
- 能够将推荐正确率提高10%者，将获得100万美元的奖励

竞赛数据

- 训练数据集：由Netflix.com提供的由来自超过480K名随机选择的匿名用户对接近18K部电影的影评（超过100M条）
- 影评的等级范围：从★到★★★★★
- 测试数据集：包含超过2.8M条记录。每条记录包含用户ID、电影ID、日期、影评等级，但是影评信息是空白。测试数据集是训练数据集的一个子集。

竞赛数据

- 有17770个影片的影评文件，每个文件代表一部影片，描述了观众对于该影片的评价

- 评价的指数从★到★★★★★，每个影评文件的格式如：
  - 第一行：movieID
  - 其余每行：userID, rating, date

- 两个影评文件中的观众id可能有相同的，也可能不同。

对提交算法的要求

- 对于测试数据集中的任一个<用户ID, 电影ID>对，算法必须能够预测出该用户对该电影的影评，即给出影评的星级(1-5级)。

- 算法的评价标准
  - 测试集将被划分为不交的两个子集（划分的方法是保密的），对这两个子集中的每一个预测影评，Netflix.com将计算其与对应的真实影评的均方根误差(RMSE, Root Mean Squared Error)，计算结果精确到0.0001。
  - $\delta = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n-1}}$ （这里n是测量次数，di是一组测量值与平均值的误差。）

竞赛结果

- Cinematch影片推荐引擎形成了在前述训练数据集上的预测算法，该预测算法对测试数据集所做的预测的RMSE为0.9525。
- 要想拿到百万美金大奖，参赛者至少将预测的精确度提高10%，所以，算法的预测结果的RMSE必须不大于0.8572。
- BellKor's Pragmatic Chaos 为本次竞赛的领先团队，他们所提交的算法的RMSE为0.8567。

Netflix中的聚类问题

- 目的：根据观众的评价对这17770部影片进行数据挖掘，输出约400个聚类，使得每个聚类中的影片是相似的

- 考虑：
- 1. 怎样定义及计算这类聚类问题中的相似度
- 2. 怎样表示一个聚类（或聚类中心）
- 3. 对于高维数据怎样预处理
- 4. 怎样减少高维数据的计算量

参考文献

- *Web Mining – II: Parallelizing K-Means Clustering with MapReduce.* By Tushar Deshpande , Tejas Vora .
- *Mining of Massive Datasets.* By Anand Rajaraman, Jeffrey D. Ullman.
- *Data-Intensive Text Processing with MapReduce.* By Jimmy Lin and Chris Dyer
- *Data Algorithms – Recipes for Scaling Up with Hadoop and Spark.* By Mahmoud Parsian

THANK YOU

南京大学
NANJING UNIVERSITY

南京大学计算机软件研究所
Institute of Computer Software, Nanjing University