

IRT理论

认为被试者在测验项目上的反应和成绩与他们的潜在特质有特殊的关系。**潜在特质**是在观察分析测验反应基础上提出的一种统计构想，在测验中一般是指潜在的能力，并经常用**测验总分**作为这种潜力的估算。通过项目反应理论建立的项目参数具有恒久性的特点，意味着不同测量量表的分数可以统一。

IRT模型的三条基本假设

- 能力单维性假设：指对组成某个测验的所有项目都是测量同一潜在特质
- 局部独立性假设：对某个被试而言，项目间无相关存在。用数学语言描述：给点潜在特征 θ ，项目的回答被认为是条件独立的，并且遵循某种分布 $p(\theta) = Pr(X_j = 1|\theta)$ ，函数 $p(\theta)$ 叫做项目反应函数，项目特征曲线或者是项目曲线。一般随 θ 单调递增，所以被测者能力水平越高，那么回答对项目的概率越高。
- 项目特征曲线假设：指对被试某项目的正确反应概率与能力之间的函数关系所作的模型

IRT最大的优点在于**题目参数的不变性**，（这里，矩阵分解的itemFeature矩阵也是不变的），也就是说被试在某一试题上的成绩不受他在测验中其他试题上的成绩影响；同时，在试题上的各个被试的作答也是彼此独立的，尽由各被试的潜在特质水平影响。即所谓局部独立性假设。

IRT 常用模型

IRT根据受测者回答问题的情况，通过对题目特征函数的运算，来推测受测者的能力。其，题目参数有，

- a. 难度(difficulty index)
- b. 区分度(discriminative powder index), 它的值越大说明对受测者的区分程度越高
- c. 猜测系数(guessing index), 它的值越大说明不论受测者能力高低，都容易猜对。其直观意义，即当一个用户的能力值非常低（比如负无穷）的时候，但是他仍然有可能做对这道题的概率。

三参数logistic IRT模型

$$P(\theta) = c + \frac{1-c}{1+e^{-Da(\theta-b)}}, \text{ 其中D为常数1.7}$$

应用过程中可能存在的问题，如何定义猜对这个概念？

Rasch单参数模型

如果是0/1评分，被试在某个项目上获得分数的概率可以表示为

$$P(\theta) = \frac{e^{\theta-b}}{1+e^{\theta-b}}$$

转换为对数形式

$$\log \frac{P(\theta)}{1-P(\theta)} = \theta - b$$

特定的个体对特定的题目做出特定反应的概率可以用个体能力与该题目难度的一个简单函数来表示。Rasch模型对于客观测量有两个要求，即

1. 对任何题目，能力高的个体应该比能力低的个体有更大可能作出正确回答
2. 任何个体在容易题目上的表现应该始终好过在困难题目上的表现。

Rasch模型是一个相对简单的模型，以一种最有效率的方式规定了可观测量所需要满足的条件，因此具有比较大的实践意义。

缺陷：

IRT模型应对的都是0/1模型，即只评判被测者答题的对错，但是我们系统里的分数是连续值，并不是0/1表示的。

ALS矩阵分解

ALS的证明以及推导不再赘述，通过spark进行ALS进行矩阵分解，我们可以得到Item的feature矩阵以及User的feature矩阵。

与传统的推荐问题不同，在推荐问题中，我们通过 P_i 和 u_j 的计算分别得出用户对每一个item的分数，然后根据分数从高到低列表进行Top N推荐。我们的目标并不是推荐某一个sentence给用户去学习，而是希望衡量用户当前的水平。

初始💡

对应前述Rosch模型，题目难度 b 是固定不变的，而在我们的user-sentence(maybe word or ngram)矩阵中Item的feature矩阵也一样是不变的。也就是说，我们可以把不同item所对应的feature向量，但是Rasch模型中表示的是一个0到1的概率值，并不适用于连续值的处理情况。

进一步💡：KDD Cup 2010 Challenge

与传统的评分预测问题不同的是，学生的水平随着练习其实是在发生变化的。

我们来看[KDDCUP2010 challenge](#)的问题：

How generally or narrowly do students learn? How quickly or slowly? Will the rate of improvement vary between students? What does it mean for one problem to be similar to another? It might depend on whether the knowledge required for one problem is the same as the knowledge required for another. But is it possible to infer

the knowledge requirements of problems directly from student performance data, without human analysis of the tasks?

不过和Kdd Cup2010 challenge 问题中，providing **Correct First Attempt** values for the test portion.

Correct First Attempt: your prediction value, a decimal number between 0 and 1, that indicates the probability of a correct first attempt for this student-step.

Table 1. Data from the "Making Cans" example, aggregated by student-step

| Row | Student Problem | Step | Incorrects | Hints | Error Rate | Knowledge component | Opportunity Count |
|-----|----------------------|-----------------------|------------|-------|------------|---------------------|-------------------|
| 1 | So1 WATERING_VEGGIES | (WATERED-AREA Q1) | 0 | 0 | 0 | Circle-Area | 1 |
| 2 | So1 WATERING_VEGGIES | (TOTAL-GARDEN Q1) | 2 | 1 | 1 | Rectangle-Area | 1 |
| 3 | So1 WATERING_VEGGIES | (UNWATERED-AREA Q1) | 0 | 0 | 0 | Compose-Areas | 1 |
| 4 | So1 WATERING_VEGGIES | DONE | 0 | 0 | 0 | Determine-Done | 1 |
| 5 | So1 MAKING-CANS | (POG-RADIUS Q1) | 0 | 0 | 0 | Enter-Given | 1 |
| 6 | So1 MAKING-CANS | (SQUARE-BASE Q1) | 0 | 0 | 0 | Enter-Given | 2 |
| 7 | So1 MAKING-CANS | (SQUARE-AREA Q1) | 0 | 0 | 0 | Square-Area | 1 |
| 8 | So1 MAKING-CANS | (POG-AREA Q1) | 0 | 0 | 0 | Circle-Area | 2 |
| 9 | So1 MAKING-CANS | (SCRAP-METAL-AREA Q1) | 2 | 0 | 1 | Compose-Areas | 2 |
| 10 | So1 MAKING-CANS | (POG-RADIUS Q2) | 0 | 0 | 0 | Enter-Given | 3 |
| 11 | So1 MAKING-CANS | (SQUARE-BASE Q2) | 0 | 0 | 0 | Enter-Given | 4 |
| 12 | So1 MAKING-CANS | (SQUARE-AREA Q2) | 0 | 0 | 0 | Square-Area | 2 |
| 13 | So1 MAKING-CANS | (POG-AREA Q2) | 0 | 0 | 0 | Circle-Area | 3 |
| 14 | So1 MAKING-CANS | (SCRAP-METAL-AREA Q2) | 0 | 0 | 0 | Compose-Areas | 3 |
| 15 | So1 MAKING-CANS | (POG-RADIUS Q3) | 0 | 0 | 0 | Enter-Given | 5 |

在最后获胜的台大的方案中，他们用的是feature engineering，然后通过多个team的分类器结果进行集成。他们做了多组feature，然后分别用LR训练，最后再用LR进行集成。优化的目标函数最终为： $\min_w \|\mathbf{y} - P\mathbf{w}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$

缺点，我们的问题主要构成为评分，并不是步骤式的问题，同时我们没有做这么细致的feature engineering。

Factorization Models for Forecasting Student Performance

[Factorization Models for Forecasting Student Performance](#)

由于student,sentence,rating可以被看作是推荐系统中常用的三元组，因此可以转换为评分预测问题来处理，同时因为存在时间的影响，我们采用张量来表示整个模型

$$\text{即 } Z = \sum_{k=1}^K \lambda_k w_k \wedge h_k \wedge q_k$$

三个向量分别代表的是User, Item, 以及Time中的latent factors

我们通过以下公式对用户的分数进行预测，

$$p_{uiT^*} = \sum_{k=1}^K w_{uk} h_{ik} \phi_{T^*k} \text{ 其中 } \phi_{T^*k} = \frac{\sum_{t=T^*-L}^{T^*-1} q_{tk} p_t}{L}$$

where T^* is the current time in the sequence; q_{tk} and p_t are the time latent factor and the student performance

of the previous time, respectively; L is the number of steps in the history to be used by the model.

为了考虑，user bias 以及item bias,我们将上述预测的公式更改为，

$$p_{uiT^*} = \mu + b_u + b_i + \sum_{k=1}^K w_{uk} h_{ik} \phi_{T^*k}$$

其中 μ 是所有user在所有训练集中在item上的平均得分，而 b_u 则是user u 相对于 μ 的偏置分数， b_i 是item上所有user的分数减去 μ

在education领域，我们需要考虑的是知识的延续性，它和普通商业领域一环扣一环不同，user读某一句话不仅仅与之前一周的技能点有关，相反，用户读某一句话是和他过去所掌握的累积的知识相关的，也就是说我们需要考虑的是使用用户之前所有练习中的表现，因此 ϕ_{T^*k} 我们重新定义如下，

$$\phi_{T^*k} = \frac{\sum_{t=T^*-L}^{T^*-1} h'_{tk} q_{tk} p_t}{L} \text{ 其中 } h'_{tk} \text{ 表示的是上一个sentence中的隐因子向量}$$

采用Graphx实现SGD解MF

$$L = \sum_{(u,i) \in S} [(R_{ui} - (p_u * q_i))^2 + \lambda(\|p_u\|_2 + \|q_i\|_2)]$$

那么我们可以求梯度得到使得L最小的Q和P矩阵

$$\begin{aligned} \frac{\partial L}{\partial p_u} &= \sum_{i \in R(u)} [2 * (R_{ui} - p_u * q_i) * (-q_i) + 2\lambda p_u] \\ &= 2 \sum_{i \in R(u)} [(R_{ui} - p_u * q_i) * (-q_i) + \lambda p_u] \quad (3) \end{aligned}$$

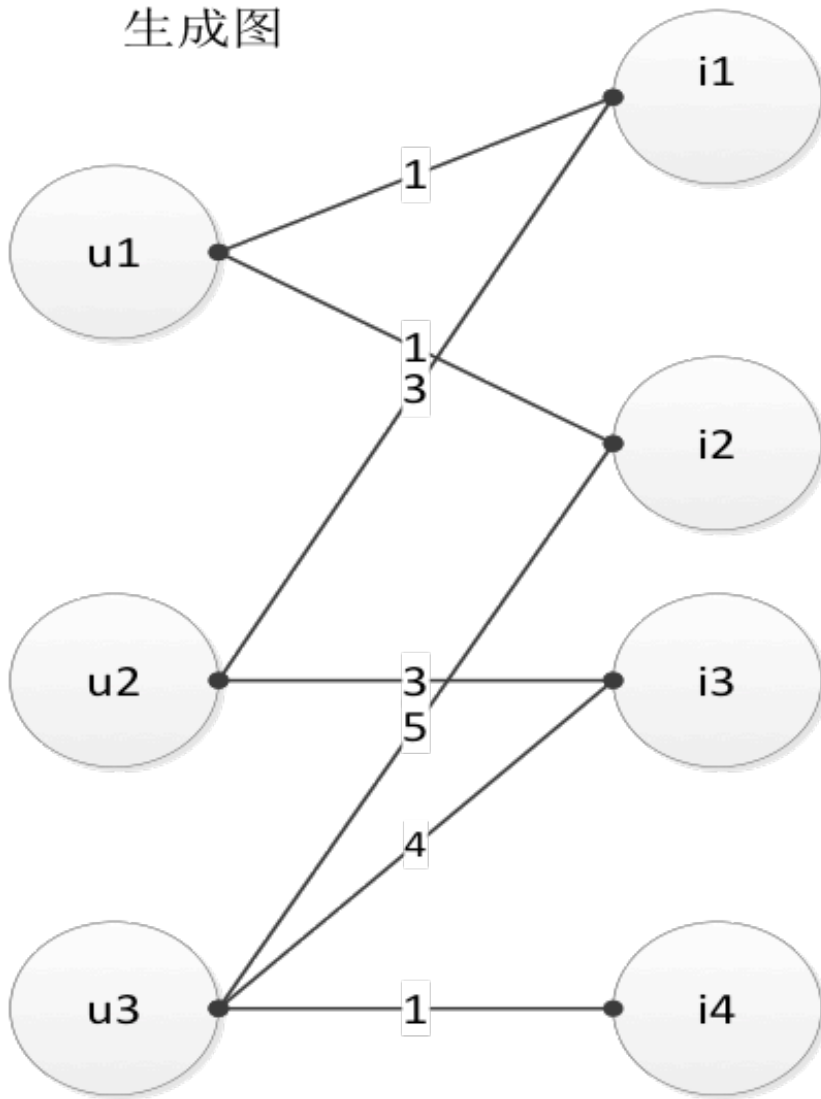
$$\frac{\partial L}{\partial q_i} = 2 \sum_{u \in R(i)} [(R_{ui} - p_u * q_i) * (-p_u) + \lambda q_i] \quad (4)$$

我们把学习率记作是 α , 则 $p_u = p_u - \alpha \frac{\partial L}{\partial p_u}$

通过二部图来实现SGD

每次当某个用户的K维向量需要更新的时候，只有这个用户打分过的物品对应的K维向量会参加运算，而和其他元素无关，而且一个用户读过的句子和库里所有的句子比只是一小部分，那么我们可以用一个二部图来表示用户的评分矩阵。

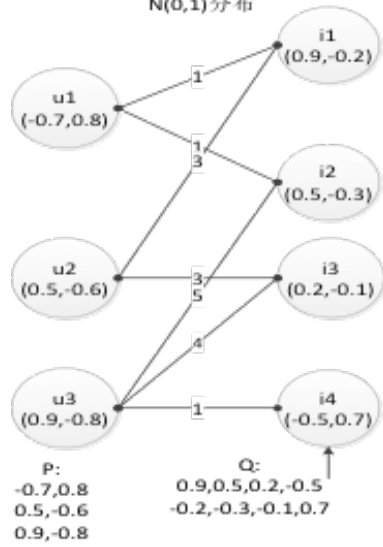
根据评分矩阵 生成图



初始化 $P: m * k, Q: k * n$

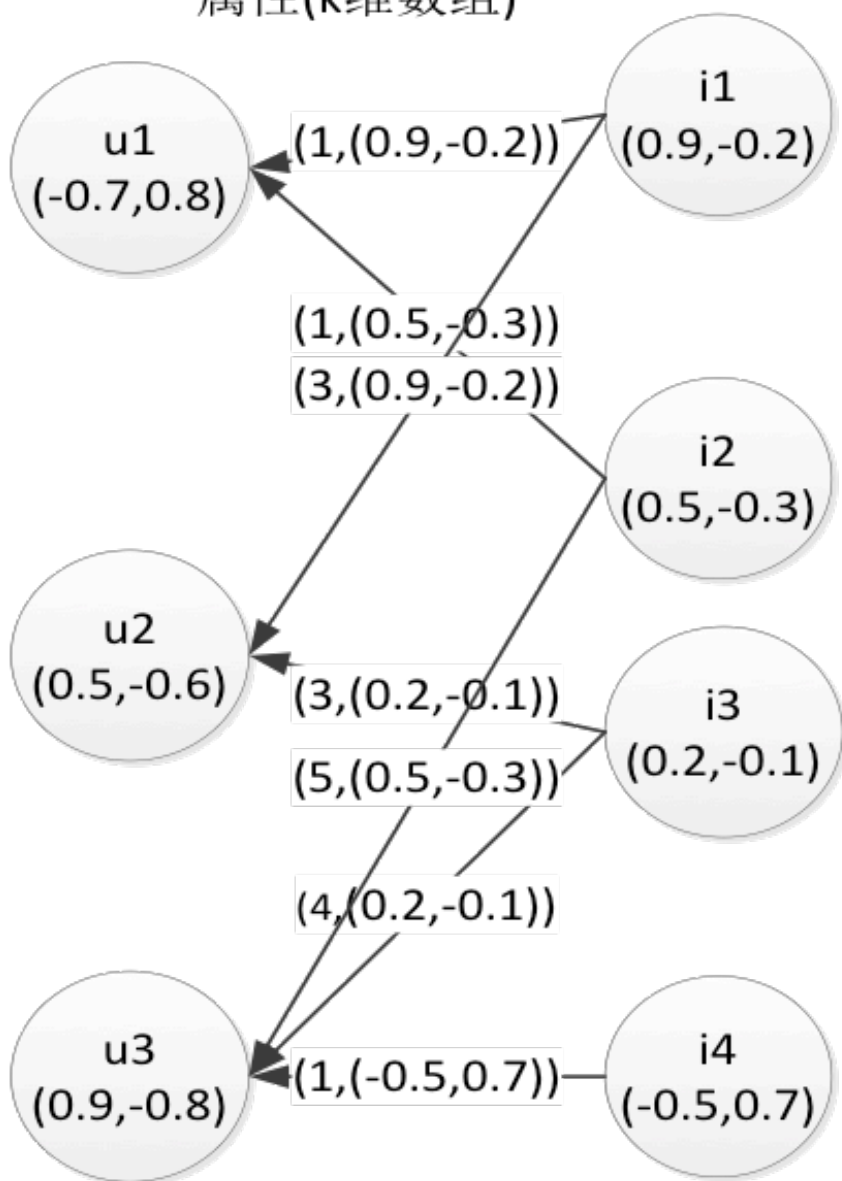
随机初始化其中元素符合正态分布 $N(0,1)$

初始化 $P:m*k,Q:k*n$ 两个矩阵,
这里 $m=3,n=4,k=2$,元素值符合
 $N(0,1)$ 分布



第一步用户得到物品边属性(评分， 物品点的属性(k 维数组))

第一步:用户点
得到边属性(评分)
和物品点的
属性(k维数组)



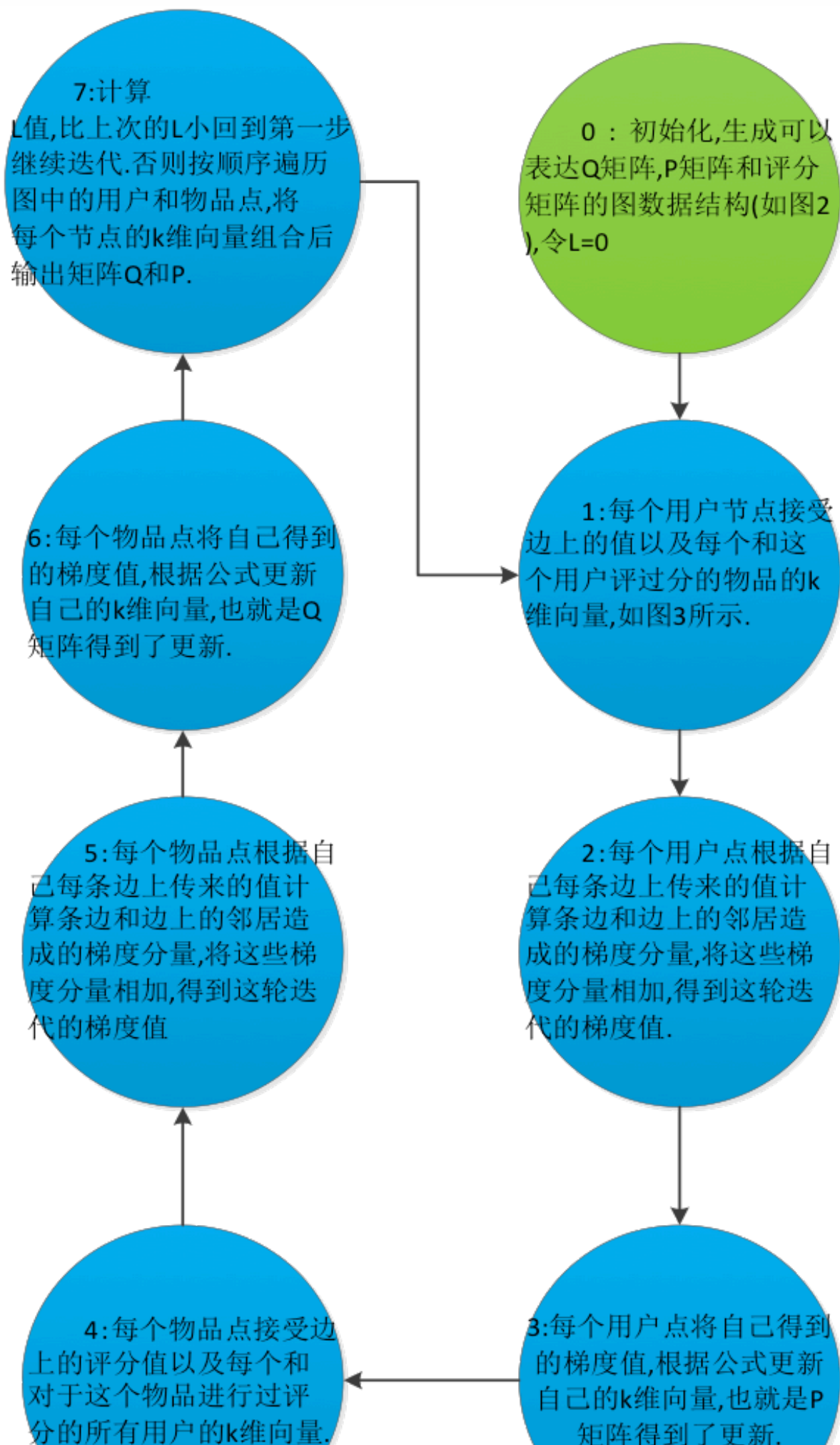
第二步的时候，用户根据自己每条边上传来的值，计算边和边上邻居造成的梯度分量，将梯度分量相加得到这一轮的梯度值


第三步，用户根据自己得到的梯度值，根据 $\mathbf{p}_u = \mathbf{p}_u - \alpha \frac{\delta L}{\delta \mathbf{p}_u}$ 更新自己的K维向量，则P矩阵被更新

第四步，每个物品点接收边上传来的评分值，以及每个读过这一句的所有用户的K维向量，计算得到这一轮的梯度值

第五步，根据 $\mathbf{q}_i = \mathbf{q}_i \alpha \frac{\phi L}{\phi \mathbf{q}_i}$ 更新自己的K维向量，即更新Q矩阵

第六步，判断L有没有比上次的L小。如果是 返回第一步重新迭代，否则退出迭代，输出Q和P





对于某些点,误差值在迭代开始没几轮的时候就达到了目标范围,这个某些点提前结束迭代的功能对于物品节点也成立.这些点可以不用参加之后的迭代,在基于spark的图计算框架graphx提供了这样功能的函数:

mapReduceTriplets(sendMsg, mergeMsg, Some((newVerts, activeDir)))

这个函数是GraphX中最核心的一个函数,表示对于图中的所有以两个点一条边组成的三元组Triplets进行一次数据传输与计算,具体过程为: sendMsg, 相当于map, 应用于每一个Triplet上, 生成一个或者多个消息, 消息以Triplet关联的两个顶点中的任意一个或两个为目标顶点; mergeMsg, 相当于reduce, 应用于每一个Vertex上, 将发送给每一个顶点的消息合并起来。前面的两个参数别是在”think like vertex”的情况下每个点对于邻居节点发出的信息sendMsg和对于自己的邻居发给自己的信息的处理函数mergeMsg.而第三个参数确定了满足一定条件的点参与这一次的图的迭代. GraphX针对这种应用场景进行了较为深层的优化 没有更新的顶点在下一轮迭代时不需要向邻居重新发送消息。因此, mapReduceTriplets遍历边时, 如果一条边邻居点值在上一轮迭代时没有更新, 则直接跳过, 避免了大量无用的计算和通信。

这一部分参考 [矩阵分解与图计算框架](#)

Continue?

后续延伸, [如何在图中加入时间信息?](#)

[graphx](#)

根据data bricks提供的slide显示, Graphx是可以实现Tensor Factorization的功能的