Computer Architecture

Parallel Programming

Project due: 10 May, 23:59pm

1 项目内容描述：

本项目内容是设计一个基于矩阵分解的协调过滤推荐算法，可用于电影推荐系统。基于电影的评分数据，大家可以用这些数据来实现一个完整的协同过滤推荐算法来帮助用户发现可能喜欢的电影。

1.1 如何预测用户给电影的评分？

在本项目中，你们根据 http://grouplens.org/datasets/movielens/ 提供的用户对电影的评分数据集来设计算法。在数据集中，评分分值范围 1-5 分，分数越高，可视为用户越喜欢这个电影。你们的任务是设计算法来预测用户对未看过电影的评分。

2 基于矩阵分解的协调过滤推荐算法

我们获得的数据可构成一个用户-物品评分矩阵，如下图所示：

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

矩阵中，描述了 5 个用户 (U1,U2,U3,U4,U5) 对 4 个物品 (D1,D2,D3,D4) 的评分 (1-5 分)，-表示没有评分，现在目的是把没有评分的给预测出来，然后按预测的分数高低，给用户进行推荐。

如何预测缺失的评分呢？对于缺失的评分，可以转化为基于机器学习的回归问题，也就是连续值的预测，对于矩阵分解有如下式子，R 是类似上图的评分矩阵，假设 $N * M$ 维 (N 表示行数，M 表示列数)，可以分解为 P 跟 Q 矩阵，其中 P 矩阵维度 $N * K$，P 矩阵维度 $M * K$：

$$\mathrm{R} \approx \mathrm{P} \times \mathrm{Q}^{T} = \hat{\mathrm{R}}$$

直观上，P 矩阵是 N 个用户对 K 个主题的关系，Q 矩阵是 K 个主题跟 M 个物品的关系。至于 K 个主题具体是什么，可由设计者给出解释（实际上是一种软聚类）。在算法里面 K 是作为可调节的参数，通常 K 在 10~100 之间。

$$\hat{r}_{ij} = \mathrm{p}_i \mathrm{q}_j^{T}$$

这里$\hat{r}_{ij}$为预测的评分矩阵$\hat{\mathrm{R}}$中第i行和第j列的元素，p_i为矩阵 P 中第i行向量，q_j为矩阵 Q 中第j行向量。

由此，矩阵分解模型将 user 和 item 映射为对应的潜在因素矩阵（latent-factor matrix) P 和 Q。每个 user 对应一个向量，每个 item 对应一个向量 ，user 和 item 之间的关系（偏好）就被建模为两个向量的内积。两种向量可以包含一定的含义，比如，将 user 的兴趣表示为 4 维向量，分别代表：

1. 对剧本的偏好
2. 对特效场面的偏好
3. 对演员的要求
4. 对导演偏好

相应地，item 对应的特点也表示为 4 维向量:

1. 剧本
2. 特效场面
3. 演员
4. 导演

最终，一一对应，user 对 item 的偏好程度自然就是两个向量的乘积了。 下面的式子给出了预测评分与实际分数的误差：

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \mathrm{p}_i\mathrm{q}_j^T)^2,$$

考虑到用户之间打分的尺度问题，如有的用户总是会打出比别人高的分，或者说有的用户他的评价尺度比较宽松；同样有的 item 总是被打高分。这是一个普遍存在的问题，所以在构造目标函数的时候需要增加几项：所有评分的平均值μ，user 的偏见分数$b1$和 item 的偏见分数$b2$：

$$\hat{r}_{ij}=\mu + b1_i + b2_j + \mathrm{p}_i\mathrm{q}_j^T,$$

这里$b1_i$和$b2_j$分别是用户 i 和物品 j 的基于打分偏差得到的基线（baseline），μ为打分的平均值，μ, $b1$和$b2$可以直接统计得到，例如：

$$\mu = \frac{1}{|\mathrm{R}|}\sum_{i,j\in\mathrm{R}} r_{ij};$$
$$b1_i = \frac{1}{|\mathrm{R}_i|}\sum_{j\in\mathrm{R}_i}(r_{ij} - \mu);$$
$$b2_j = \frac{1}{|\mathrm{R}_j|}\sum_{i\in\mathrm{R}_j}(r_{ij} - \mu).$$

这里 R 是打分矩阵，R_i是 R 中对用户 i 的向量，R_j是 R 中对物品 j 的向量。

为了避免矩阵 P 和 Q 出现病态矩阵，且更具有合理性和可解释性，我们需要增加正则项。由此，我们可以得到总的损失函数如下：

$$\min Loss(P,Q) = \sum_{(i,j)\in R} (r_{ij} - \mu - b1_i - b2_j - \mathrm{p}_i\mathrm{q}_j^T)^2 + \lambda(b1_i^2 + b2_j^2 + \left\|\mathrm{p}_i\right\|^2 + \left\|\mathrm{q}_j\right\|^2)$$

这里 $\lambda(b1_i^2 + b2_j^2 + \left\|\mathrm{p}_i\right\|^2 + \left\|\mathrm{q}_j\right\|^2)$ 是加入的正则项，λ 是正则化因子（一般手动给定）。有了目标函数，我们就可以用随机梯度下降（Stochastic Gradient Descent, SGD）来优化此函数。

对目标函数求梯度，可得迭代更新的各个元素：

$$\begin{aligned}
e_{ij} &= r_{ij} - \hat{r}_{ij} \\
b1'_i &= b1_i + \gamma(e_{ij} - \lambda b1_i) \\
b2'_j &= b2_j + \gamma(e_{ij} - \lambda b2_j) \\
\mathrm{p}'_i &= \mathrm{p}_i + \gamma(e_{ij}\mathrm{q}_j - \lambda \mathrm{p}_i) \\
\mathrm{q}'_j &= \mathrm{q}_j + \gamma(e_{ij}\mathrm{p}_i - \lambda \mathrm{q}_j)
\end{aligned}$$

这里 γ 是学习速率，需要手动指定。

3 评价机制

3.1 预测电影评分

这部分的评判标准我们采用参赛者的推荐预测评分值与实际评分值之间的均方根误差 RMSE (Root Mean Squared Error)。在划分好的测试集中，参赛者的算法预测用户对某电影的预测评分是 Y，而用户对此电影的实际评分是 X，那么所得的 RMSE = sqrt(sum(for all items(X-Y)^2)/n)。此部分评分占总分之 30%。

Matlab 算法参考：Intel Q8200 at 3.0GHz, 8G RAM, 在收敛至可接受的 RMSE 的终止迭代条件下，完成 1000*1700 的矩阵恢复需要 60s。

3.2 并行算法的加速比

这部分的评价标准是利用并行算法优化后获得的加速比，加速比定义为 R = T0 / TP，这里 TP 为并行算法使用的时间，T0 为未优化的算法使用时间。此部分评分占总分之 70%。

4 代码算法

本次 Project 允许使用任语言进行编写。

5 于指定时间前提交至 yyliang@must.edu.mo。