推荐系统中的矩阵分解

李向阳　d1142845997@gmail.com

目录

1 引入

这一次继续说推荐算法.

前面我们已经介绍了协同过滤推荐算法, 并知道了推荐算法的框架其实就是一个评分矩阵, 而我们的主要目的便是"补齐"这个评分矩阵. 既然是矩阵, 自然有很多矩阵的办法可以解决这个问题, 比如矩阵分解. 我们已经学过矩阵分析这门课, 所以理论基础不用多说, 我们直接讲推荐算法.

假设样本集中有 U 个用户 (User), 有 I 个物品 (Item), 设用户 u 对物品 i 的评分为 r_{ui}, 评分矩阵 $R = (r_{ui})_{U\times I}$. 我们知道, 评分矩阵 R 是不完整的, 我们的目的便是合理的补全这个矩阵.

2 矩阵分解模型

2.1 SVD 基本模型

设矩阵 $R_{U\times I}$ 的秩为 K, 即 $\mathrm{rank}(R) = K$, 那么由满秩分解定理知, 存在矩阵 $P_{U\times K}$ 和 $Q_{K\times I}$, 使得

$$R_{U\times I} = P_{U\times K}\cdot Q_{K\times I}$$

其中 $\mathrm{rank}(P) = \mathrm{rank}(Q) = K$.

所谓矩阵分解模型, 便是利用评分矩阵 R 的已知评分来估计矩阵 P 和 Q, 即使得 P 和 Q 相乘的结果能够最好的拟合已知的评分, 然后我们就用 P 乘 Q 的结果矩阵作为预测的评分矩阵, 那么所有未知的评分也就知道了. 或者说, 任何一个未知的就可以用 P 的某一行乘上 Q 的某一列得到了, 比如要预测用户 u 对物品 i 的评分, 那么我们可以用 P 矩阵的第 u 行乘上 Q 矩阵的第 i 列.

$$\hat{r}_{ui} = p_u^T q_i$$

这里 p_u 和 q_i 表示向量, 本应该用粗体表示, 但是为了方便书写, 而且这不像机器学习中的很多算法记号容易搞混, 本算法的记号还是很容易区分的, 所以写的就比较随意了.

总之, 我们的目的便是估计出矩阵 P 和 Q.

2.2 潜在因子模型与非负矩阵分解

其实矩阵分解的思想我们在协同推荐算法中一种接触到了, 在那里, 我们实际上不正是通过用户的相似度矩阵再乘以评分向量矩阵得到预测的评分吗?

此外, 还有一类推荐算法叫做潜在因子 (Latent Factor) 算法或者隐语义模型. 它是什么呢?

潜在因子算法也是把评分矩阵 R 满秩分解为两个矩阵 W 和 H 的乘积, 其中 $W(i,j)$ 被解释为用户 i 属于类别 j 的权重, $H(a,b)$ 被解释为物品 b 属于类别 a 的权重.

我们以音乐推荐为例具体阐述一下, 该例子来自知乎.

这种算法的思想是: 每个用户都有自己的偏好, 比如用户 u 喜欢的音乐特征有小清新、钢琴伴奏、王菲等, 这些特征我们不知道, 称为潜在因子. 如果一首歌带有这些特征, 我们就将这首歌推荐给用户 u. 也就是说, 我们是用特征 (潜在因子) 去连接用户和音乐. 每个人对不同的特征偏好不同, 而每首歌包含的特征也不一样, 我们希望找到如下两个矩阵:

(1) 用户-潜在因子矩阵 Q

其表示不同的用户对不同特征的偏好程度, 比如下表

表 1: 用户-潜在因子矩阵

用户/特征	**小清新**	**吉他伴奏**	**优雅**	**伤感**	**五月天**
张三	0.6	0.8	0.1	0.1	0.7
李四	0.1	0	0.9	0.1	0.2
王五	0.5	0.7	0.9	0.9	0

(2) 潜在因子-音乐矩阵 P

其表示每种音乐含有各种元素的成分, 比如下表中表示音乐 A 是一个偏小清新的音乐, 含有小清新这个特征的成分是 0.9, 吉他伴奏的成分是 0.1, 优雅的成分是 $0.2\cdots$

表 2: 潜在因子-音乐矩阵

音乐/特征	**小清新**	**吉他伴奏**	**优雅**	**伤感**	**五月天**
音乐 A	0.9	0.1	0.2	0.4	0
音乐 B	0.5	0.6	0.1	0.9	1
音乐 C	0.1	0.2	0.5	0.1	0
音乐 D	0	0.6	0.1	0.2	0

利用这两个矩阵, 我们就可以得出张三对音乐 A 的预测评分是: 张三对小清新的偏好程度 * 音乐 A 含有小清新的成分 + 张三对吉他伴奏的偏好

程度 * 音乐 A 含有吉他伴奏的成分 + 张三对优雅的偏好程度 * 音乐 A 含有优雅的成分 + $\cdots$, 如下

$$0.6 \times 0.9 + 0.8 \times 0.1 + 0.1 \times 0.2 + 0.1 \times 0.4 + 0.7 \times 0 = 0.68$$

每个人对任何一首歌的预期评分都可以这样计算, 最终可得预测评分矩阵

表 3: 预测评分矩阵

用户 / 音乐	**音乐 A**	**音乐 B**	**音乐 C**	**音乐 D**
张三	0.68	1.58	0.28	0.51
李四	0.31	0.43	0.47	0.11
王五	1.06	1.57	0.73	0.69

实际上就是

$$\hat{R} = Q \cdot P^T$$

由预测评分矩阵可知, 我们应该给张三推荐音乐 B, 对李四推荐音乐 C, 对王五推荐音乐 B.

这就是潜在因子模型的一个具体的解释. 可以看到, 关键还是要得到矩阵 P 和 Q, 下面我们就回到矩阵分解的模型, 来看参数估计.

2.3 参数估计

如何估计矩阵 $P = (p_{uk})$ 和 $Q = (q_{ki})$ 呢?

我们还是采用最经典的最小二乘估计, 即真实值与预测值的误差为

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

我们把总的误差平方和作为损失函数:

$$\mathcal{L} = \frac{1}{2} SSE = \frac{1}{2} \sum_{u,i} e_{ui}^2 = \frac{1}{2} \sum_{u,i} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2$$

上式中的求和是针对样本集中那些非零的 r_{ui} 的.

具体计算, 还是老办法, 即梯度下降法.

我们要估计的是矩阵 P 和 Q 的所有元素 p_{uk} 和 $q_{ki}, u = 1, 2, \cdots, U, k = 1, 2, \cdots, K, i = 1, 2, \cdots, I$.

下面来求导数, 先看单个样本, 即 $SSE = e_{ui}^2$, 可得

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_{uk}} &= \frac{1}{2} \cdot 2e_{ui} \cdot \frac{\partial e_{ui}}{\partial p_{uk}} \\ &= e_{ui} \cdot (-q_{ki}) \\ &= -e_{ui}q_{ki}\end{aligned}$$

同理可得

$$\frac{\partial \mathcal{L}}{\partial q_{ki}} = -e_{ui}p_{uk}$$

因此, 随机梯度下降的更新公式为

$$\begin{aligned}p_{uk} &:= p_{uk} - \eta(-e_{ui}q_{ui}) = p_{uk} + \eta e_{ui}q_{ki} \\ q_{ki} &:= q_{ki} - \eta(-e_{ui}p_{uk}) = q_{ki} + \eta e_{ui}p_{uk}\end{aligned}$$

相应的, 批梯度下降的更新公式为

$$\begin{aligned}p_{uk} &:= p_{uk} + \eta \sum_{u,i} e_{ui}q_{ki} \\ q_{ki} &:= q_{ki} + \eta \sum_{u,i} e_{ui}p_{uk}\end{aligned}$$

其中 η 为步长, 也称学习速率, 关于随机梯度下降 (SVD) 与批梯度下降, 已经在机器学习中线性回归中提过了. 随机性可以带来很多好处, 比如有利于避免局部最优解, 所以一般使用随机梯度下降法.

利用更新式不断迭代到收敛, 就可以得到我们想要估计的参数了.

2.4 正则化

同机器学习中的很多模型一样, 矩阵分解模型也可能过拟合, 所以要对参数施加惩罚来正则化, 由于矩阵 P 和 Q 中的所有元素都是变量, 所以对它们都施加惩罚, 即构造损失函数如下:

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda \sum_{u} ||p_u||^2 + \frac{1}{2}\lambda \sum_{i} ||q_i||^2 \\ &= \frac{1}{2}\sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda \sum_{u=1}^{U}\sum_{k=1}^{K} p_{uk}^2 + \frac{1}{2}\lambda \sum_{i=1}^{I}\sum_{k=1}^{K} q_{ki}^2\end{aligned}$$

其中 λ 为惩罚参数, 对单个样本计算导数可得

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial p_{uk}} &= -e_{ui}q_{ki} + \lambda p_{uk} \\ \frac{\partial \mathcal{L}}{\partial q_{ik}} &= -e_{ui}p_{uk} + \lambda q_{ik}\end{aligned}$$

因此, 正则化后的随机梯度下降 (RSVD) 更新公式为

$$p_{uk} := p_{uk} + \eta(e_{ui}q_{ki} - \lambda p_{uk})$$
$$q_{ki} := q_{ki} + \eta(e_{ui}p_{uk} - \lambda q_{ik})$$

以上计算都还是非常简单的 (相对于机器学习中的很多模型来说), 不断迭代便可以得到我们的解了.

3 矩阵分解模型的推广

关于矩阵分解推荐算法的变种太多了, 叫法也不统一, 在预测的式子上加点参数 (又称为偏置) 又会出来一个名称, 下面简要介绍几个变种.

我们知道矩阵分解基本模型的预测式子如下

$$\hat{r}_{ui} = p_u^T q_i$$

以下方法都是从这个预测式子上做推广的.

3.1 RSVD 模型

这里的 RSVD 不是指正则化的随机梯度下降, 而是改进 (修正) 后的 SVD 分解模型.

由于用户对物品的评分不仅取决于用户和商品间的某种关系, 还取决于用户和商品独有的性质, Koren 把基本 SVD 的预测公式改为

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i$$

其中第一项为总的平均分, b_u 为用户 u 的属性值, b_i 为商品 i 的属性值. 也就是说, 模型中不再只有矩阵 P 和 Q 两个参数, 还加入了 b_u 和 b_i 两个变量, 也是需要求解的参数.

新加入的参数同样需要惩罚, 因此损失函数变为

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\sum_{u,i} e_{ui}^2 + \frac{1}{2}\lambda\sum_u ||p_u||^2 + \frac{1}{2}\lambda\sum_i ||q_i||^2 + \frac{1}{2}\lambda\sum_u b_u^2 + \frac{1}{2}\sum_i b_i^2 \\ &= \frac{1}{2}\sum_{u,i}\left(r_{ui} - \mu - b_u - b_i - \sum_{k=1}^{K} p_{uk}q_{ki}\right)^2 + \frac{1}{2}\lambda\sum_u ||p_u||^2 + \frac{1}{2}\lambda\sum_i ||q_i||^2 \\ &\quad + \frac{1}{2}\lambda\sum_u b_u^2 + \frac{1}{2}\sum_i b_i^2\end{aligned}$$

可以看出, 损失函数 $\mathcal{L}$ 对 p_{uk} 和 q_{ik} 的偏导数都没有变化, 不过此时模型中多了 b_u 和 b_i 参数, 我们同样需要求出其更新式. 还是以单个样本计算, 可得

$$\frac{\partial \mathcal{L}}{\partial b_u} = -e_{ui} + \lambda b_u$$
$$\frac{\partial \mathcal{L}}{\partial b_i} = -e_{ui} + \lambda b_i$$

因此其随机梯度下降更新式为

$$b_u := b_u + \eta(e_{ui} - \lambda b_u)$$
$$b_i := b_i + \eta(e_{ui} - \lambda b_i)$$

有了参数更新式, 就可以不断迭代求解了.

3.2 ASVD 模型

全称叫 Asymmetric-SVD, 即非对称 SVD.

我们说推荐算法的大致框架是评分矩阵, 不过, 有时, 人们可能只是浏览过某个商品但没有评分 (比如说我经常干这种事, 主要是懒), 而用户的浏览记录本身就能反映用户的喜好. 所以除了评分矩阵以外, 如果能考虑到浏览记录, 模型的预测性能更好.

本篇文章主要参考自博客http://blog.csdn.net/zhongkejingwang/article/details/43083603, 不过看到这个算法时感觉记号没有讲清楚, 我也没有深究, 但是先暂时记录了下来.

ASVD 的预测式子为

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - \mu - b_u - b_j) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

其中 $R(u)$ 表示用户 u 评过分的物品集合, $N(u)$ 表示用户 u 浏览过但没有评过分的物品集合, x_i 和 y_j 是商品的属性 (感觉对这两个变量没有说清楚, 不过注意它们是个向量).

可以看到, 该模型去掉了用户矩阵 P, 同时加入了一些新的参数, 相应的构造损失函数如下

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2} \sum_{u,i} e_{ui}^2 + \frac{1}{2} \lambda \sum_{j \in R(u)} ||x_j||^2 + \frac{1}{2} \lambda \sum_{j \in N(u)} ||y_j||^2 + \frac{1}{2} \lambda \sum_i ||q_i||^2 + \frac{1}{2} \lambda \sum_u b_u^2 + \frac{1}{2} \lambda \sum_i b_i^2 \\
&= \frac{1}{2} \sum_{u,i} \left(r_{ui} - \mu - b_u - b_i - \sum_{m=1}^{F} z_m q_{mi} \right)^2 + \frac{1}{2} \lambda \sum_{j \in R(u)} ||x_j||^2 + \frac{1}{2} \lambda \sum_{j \in N(u)} ||y_j||^2 \\
&\quad + \frac{1}{2} \lambda \sum_i ||q_i||^2 + \frac{1}{2} \sum_u b_u^2 + \frac{1}{2} \lambda \sum_i b_i^2
\end{aligned}$$

其中的向量 z 等于下式

$$z = |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - \mu - b_u - b_j)x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$$

下面来求 q_{ik}, x, y 的梯度下降更新式. 其实这里的 z 相当于前面的 p_u, 因此可得 q_{ik} 的更新式如下

$$q_{ki} := q_{ki} + \eta(e_{ui}z_k - \lambda q_{ik})$$

再来求 x_j 的偏导数, $\mathcal{L}$ 的第二项对 x_j 的导数很容易得到, 而第一项对 x_j 的导数为

$$\begin{aligned}\frac{\partial}{\partial x_{jk}}\left(\frac{1}{2}\sum_{u,i} e_{ui}^2\right) &= \sum_{i,j \in R(u)} \left(e_{ui}\frac{\partial e_{ui}}{\partial x_{jk}}\right) \\ &= \sum_{i,j \in R(u)} \left(e_{ui}\frac{\partial}{\partial x_{jk}}\left(r_{ui} - \mu - b_u - b_i - \sum_{m=1}^{F} z_m q_{mi}\right)\right)\end{aligned}$$

上式中求和符号中的 i, j 都是属于 $R(u)$ 的, 可以看到 z_m 只有当 $m = k$ 时才与 x_{jk} 有关, 所以

$$\frac{\partial}{\partial x_{jk}}\left(r_{ui} - \mu - b_u - b_i - \sum_{m=1}^{F} z_m q_{mi}\right) = -q_{ki}\frac{\partial z_k}{\partial x_{jk}}$$

又因为

$$z_k = |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - \mu - b_u - b_j)x_{jk} + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_{jk}$$

所以

$$\frac{\partial z_k}{\partial x_{jk}} = |R(u)|^{-\frac{1}{2}}(r_{uj} - \mu - b_u - b_j)$$

综上得到 $\mathcal{L}$ 对 x_{jk} 的导数为

$$\frac{\partial \mathcal{L}}{\partial x_{jk}} = -\left(\sum_{j,i \in R(u)} \left(e_{ui}q_{ki}|R(u)|^{-\frac{1}{2}}(r_{uj} - \mu - b_u - b_j)\right)\right) + \lambda x_{jk}$$

同理可得 $\mathcal{L}$ 对 y_{jk} 的导数为

$$\frac{\partial \mathcal{L}}{\partial y_{jk}} = -\left(\sum_{j \in N(u), i \in R(u)} \left(e_{ui}q_{ki}|N(u)|^{-\frac{1}{2}}\right)\right) + \lambda y_{jk}$$

这样就可以得到 x_{jk} 和 y_{jk} 的更新方程

$$x_{jk} := x_{jk} + \eta \left[\left(\sum_{j,i \in R(u)} \left(e_{ui} q_{ki} |R(u)|^{-\frac{1}{2}} (r_{uj} - \mu - b_u - b_j) \right) \right) - \lambda x_{jk} \right]$$

$$y_{jk} := y_{jk} + \eta \left[\left(\sum_{j \in N(u), i \in R(u)} \left(e_{ui} q_{ki} |N(u)|^{-\frac{1}{2}} \right) \right) - \lambda y_{jk} \right]$$

以上就是 ASVD 。。。。。。。

关于矩阵分解模型的推广还有很多, 这里就不多做介绍了.

4 总结

4.1 参考资料

(1) 博客:http://blog.csdn.net/zhongkejingwang/article/details/43083603, 讲的很清楚, 还有详细的推导, 该人的其它博客有些也不错.

(2) 博客:http://www.letiantian.me/2015-05-25-nmf-svd-recommend/, 提到了隐语义模型, 并有 Python 代码解决问题的实例.

(3) 知乎的一个回答:https://www.zhihu.com/question/26743347, 非常直观的解释了潜在因子模型.

参考文献

[1] 李荣华. 偏微分方程数值解法. 高等教育出版社 (2010)

[2] Zhilin Li,Zhonghua Qiao,Tao Tang.*Numerical Solutions of Partial Differential Equations-An Introduction to Finite Difference and Finite Element Methods.*(2011)

[3] 孙志忠. 偏微分方程数值解法. 科学出版社 (2011)

[4] 陆金甫关治. 偏微分方程数值解法. 清华大学出版社 (2004)

附录