中间件实现技术课程作业——电影推荐服务

Author:施金泉　2019104257

1.概述：

本project实现了一个web服务，基于深度学习实现电影推荐

- 数据集:movielens
- 推荐算法:神经协同过滤(neural collaborative filtering)
- 深度学习框架:pytorch
- 开发语言:python 3.7
- web服务框架:django
- 数据库:mongodb

2.流程图

3.推荐算法介绍

3.1 相关论文

新加坡国立大学 何向南 团队的论文

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu and Tat-Seng Chua (2017). Neural Collaborative Filtering. In Proceedings of WWW '17, Perth, Australia, April 03-07, 2017.

https://arxiv.org/abs/1708.05031

https://github.com/hexiangnan/neural_collaborative_filtering

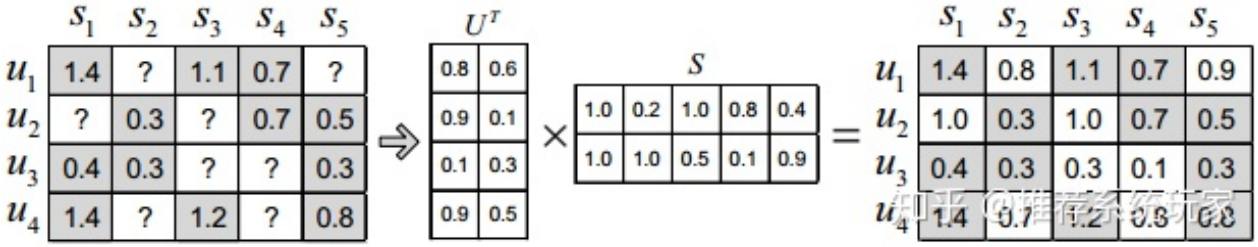
3.2 协同过滤

使用用户user与item之间的交互记录，计算用户之间相似度，用相似用户的对某个item的评分，预测某个用户对某个item的评分

	m_id_1	m_id_2	m_id_3	m_id_4	m_id_5	---	m_id_10w
u_id_1	1	1	1				
u_id_2	1	1	1	1			
u_id_3	1	1	1		1		

---							
u_id_100w	1		1		1		

3.3 矩阵分解

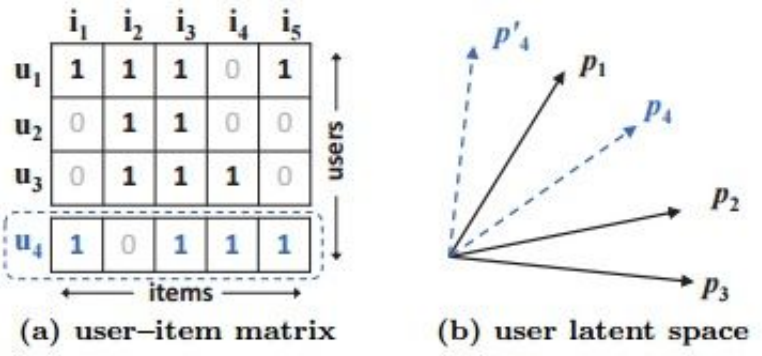


矩阵分解(MF,Matrix Factorization)，为每个user和item找到一个隐向量，问题变为：

$$\hat{y}_{u,i} = f(u, i|\mathbf{p_u}, \mathbf{q_i}) = \mathbf{p_u^T q_i} = \sum_{k=1}^{K} p_{uk} q_{ik}.$$

这里的 K表示隐式空间（latent space）的维度。正如我们所看到的，MF模型是用户和项目的潜在因素的双向互动，它假设潜在空间的每一维都是相互独立的并且用相同的权重将它们线性结合。因此，MF可视为隐向量（latent factor）的线性模型。

论文中给出了一个例子来说明这种算法的局限性：



(a) user–item matrix (b) user latent space

Figure 1: An example illustrates MF's limitation. From data matrix (a), u_4 is most similar to u_1, followed by u_3, and lastly u_2. However in the latent space (b), placing $\mathbf{p}_4$ closest to $\mathbf{p}_1$ makes $\mathbf{p}_4$ closer to $\mathbf{p}_2$ than $\mathbf{p}_3$, incurring a large ranking loss.

1(a)是user-item交互矩阵，1(b)是用户的隐式空间，论文中强调了两点来理解这张图片：
1）MF将user和item分布到同样的隐式空间中，那么两个用户之间的相似性也可以用二者在隐式空间中的向量夹角来确定。
2）使用Jaccard系数来作为真实的用户相似性。
通过MF计算的相似性与Jaccard系数计算的相似性也可以用来评判MF的性能。我们先来看看Jaccard系数

令 R_u 表示与用户 u 交互的项目集，那么用户 u 和 j 之间的Jaccard相似系数就被定义为：

$$s_{ij} = \frac{|R_i| \cap |R_j|}{|R_i| \cup |R_j|}$$

我们首先关注的图1(a)中的前三行（用户）。很容易可以计算出 $s_{23}(0.66) > s_{12}(0.5) > s_{13}(0.4)$ 。这样，$\mathbf{p}_1$，$\mathbf{p}_2$ 和 $\mathbf{p}_3$ 在潜在空间中的几何关系可绘制成图1(b)。现在，让我们考虑一个新的用户 u_4，它的输入在图1(a)中的用虚线框出。我们同样可以计算出 $s_{41}(0.6) > s_{43}(0.4) > s_{42}(0.2)$，表示 u_4 最接近 u_1，接着是 u_3，最后是 u_2。然而，如果MF模型将 $\mathbf{p}_4$ 放在了 最接近 $\mathbf{p}_1$ 的位置（图1(b) 中的虚线展示了两种不同的摆放 $\mathbf{p}_4$ 的方式，结果一样），那么会使得 $\mathbf{p}_4$ 相比与 $\mathbf{p}_3$ 更接近于 $\mathbf{p}_2$ ***（显然，根据图1(a)，u_4 应该更接近 u_3）***，这会导致很大的排名误差（ranking loss）。

上面的示例显示了MF因为使用一个简单的和固定的内积，来估计在低维潜在空间中用户-项目的复杂交互，从而所可能造成的限制。解决该问题的方法之一是使用大量的潜在因子 K (就是隐式空间向量的维度)。然而这可能对模型的泛化能力产生不利的影响（e.g. 数据的过拟合问题），特别是在稀疏的集合上。论文通过使用DNN从数据中学习交互函数，突破了这个限制。

3.4 NCF

本文先提出了一种通用框架：

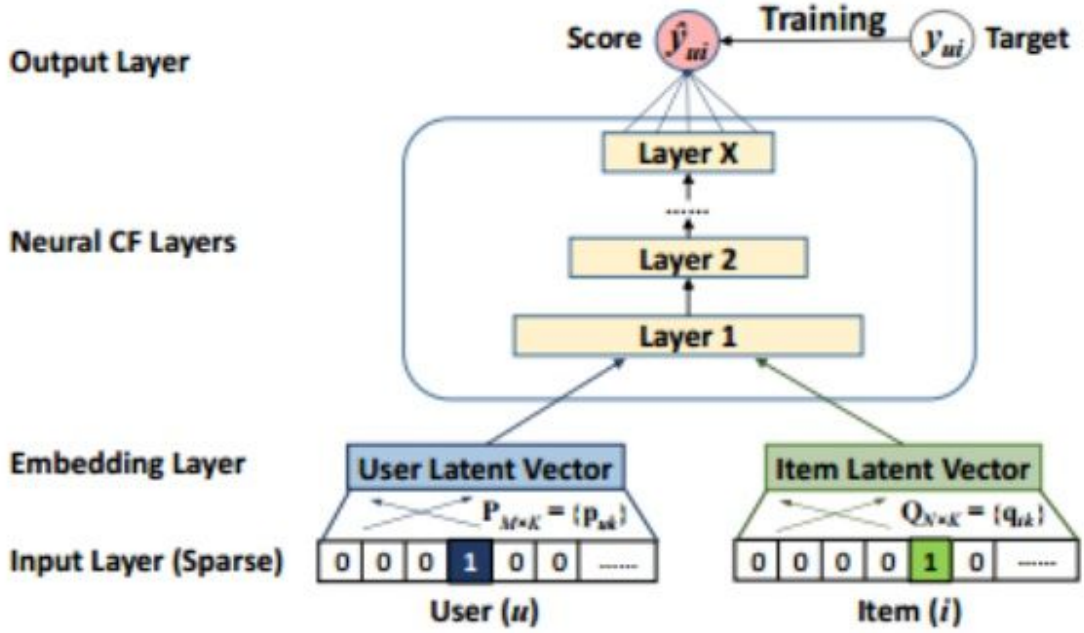


Figure 2: Neural collaborative filtering framework

针对这个通用框架，论文提出了三种不同的实现，三种实现可以用一张图来说明：

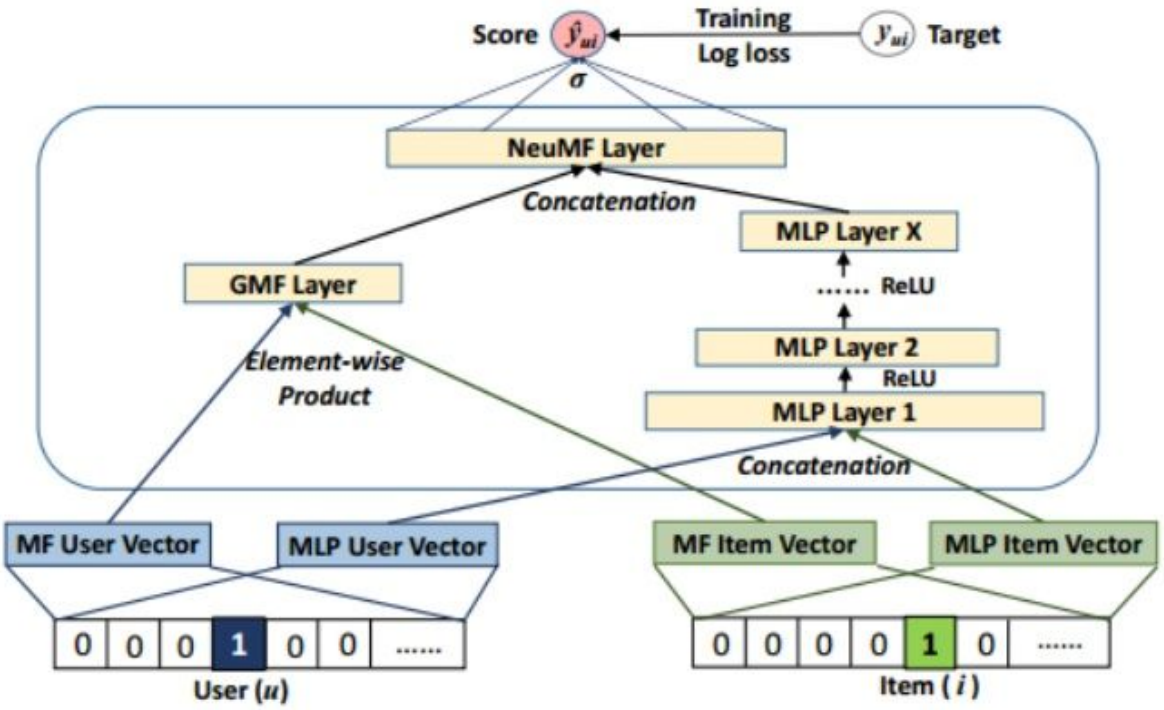


Figure 3: Neural matrix factorization model

4.部署与运行

4.1 django

demo项目，未使用nginx等服务器程序，直接run 即可运行

python manage.py runserver

调用接口

http://127.0.0.1:8000/getRecMovie/?uid=17

4.2 mongodb

```
安装MongoDB并配置如下
mongodb.conf #端口口号
port = 27017 #数据目目录
dbpath = /usr/local/mongodb/data/db
 \#日志目目录
 logpath = /usr/local/mongodb/data/logs/mongodb.log
 #设置后台运行行行
 fork = true
 \#日志输出方方式
 logappend = true
 \#开启认证
 \#auth = true
```

4.3 训练

run NCF-master下的main.py即可

4.4 预测

run NCF-master下的predict.py即可

4.5 输出训练结果到mongodb

run NCF-master下的mongodb_util.py即可

5.可优化项目

5.1 流程

使用产业界成熟的方案（spark等），实现更自动化的更新输入数据和模型更新，

5.2 代码质量

5.3 embedding的预训练与模型调参