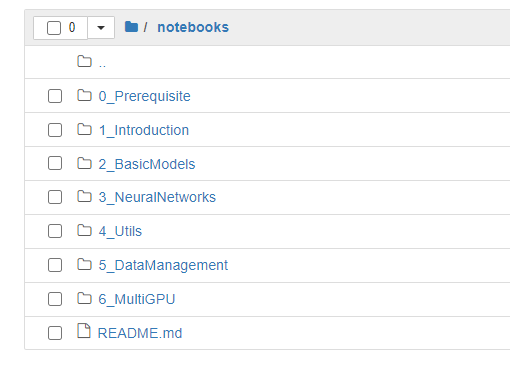
**学习周报**

1. 学习内容与进度
2. **补充学习tensorflow的知识**

（1）学习资料：github上的tensorflow-examples项目



1. 学习情况

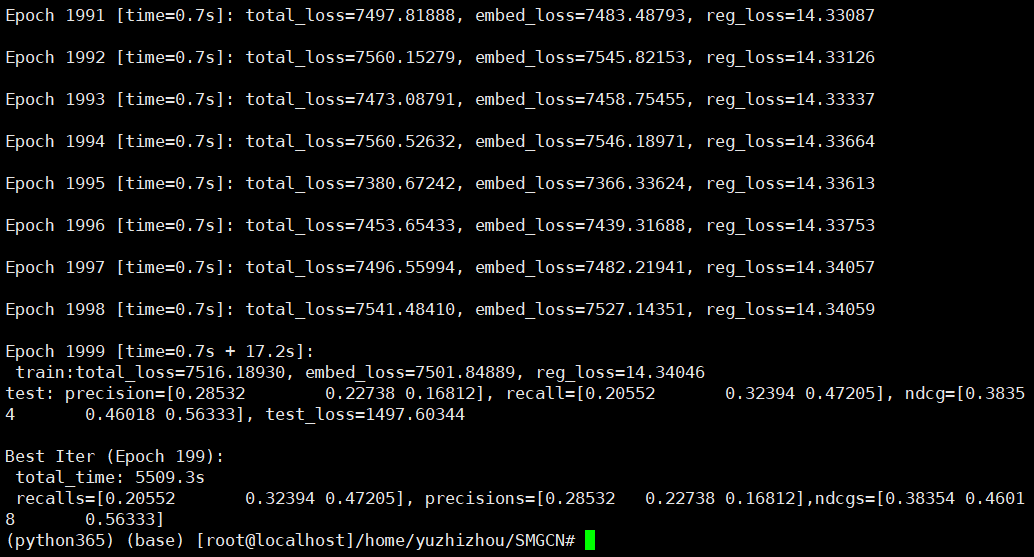
|  |  |
| --- | --- |
| Autoencoder 以及variational\_autoencoder | 使用mnist数据，包含encoder和decoder部分，encoder包含两个非线性转化层，将输入维度从h1变化为h2, decoder为同一结构，但是将维度又转变回了输入维度；利用此模型进行输入图片的视觉重建  包含三个部分，一是使用一层非线性转化层的encoder层，二是使用2层非线性转化层的decoder模型，三是中间的两层z\_mean,z\_std;  图片经过encoder层编码，然后分别馈入z\_mean和z\_std中并将结果进行线性组合，最后送入到decoder层中。 |
| Tensorflow的数据api | 使用tf.image的相关API可以对图片进行随机左右翻转，能调整亮度、饱和度、色调和对比度，以及进行随机裁剪；  使用tf.data.Dataset的相关API可以将ndarray转化为tensor数据库，并可以对数据库进行repeat、shuffle、prefetch等操作 |
| 多GPU运行 | 使用with tf.device('/cpu:0’ or ‘/gpu:0’):或者tf.Graph().device来控制具体计算所用的设备；  Tf上多GPU工作的原理是每个GPU上都保存模型，然后每块GPU单独处理数据并计算梯度，这些梯度最终汇总到CPU上并进行平均，最后在GPU上进行异步更新； |
| 参数的初始化 | 可以使用nn.Embedding来进行初始化；tf.set\_random\_seed()对数据图的全局随机种子进行设置，使得不同session中的random函数表现出相似特征； |
| Tensorboard的初级和高级使用 | 使用name\_scope使得最后的可视化结果更加清晰；  通常使用histogram来展示模型参数的梯度；  使用tf.summary.merge\_all() 来合并之前的scalar等操作，其只相当于运算符，只有在被run时才能有具体结果；  使用FileWriter和add\_summary来保存文件至磁盘； |
| 计算图 | 使用tf.reset\_default\_graph来清空计算图； tf.graph().as\_default返回一个使用此图作为默认图的上下文管理器； |

1. **根据最近学习的代码，总结下模型测试和训练的几种模式：**

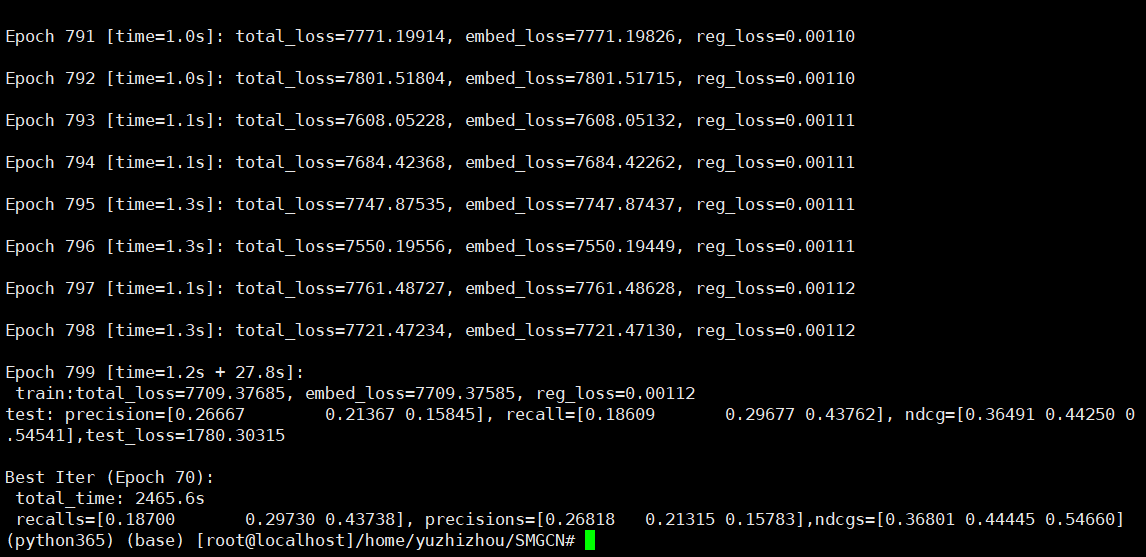
|  |  |  |
| --- | --- | --- |
| 模式 | 优点 | 缺点 |
| 训练集上训练n次，最后在测试集上只测试1次；  训练时根据迭代次数epoch间隔保存模型参数文件；测试时使用最后一次保存的参数文件进行 | 对于测试数据集很大或者测试很消耗时间的模型，比较适用 | 非常有可能得不到最优的模型参数 |
| 每训练完1次就进行1次测试；  根据测试结果保存获得最优结果的模型参数； | 对于测试数据集很小或者测试不费时间的模型，比较适用；  能够获得最优模型参数 | 相邻迭代次数间模型的参数以及效果都可能变化不大，所以频繁测试会消耗不必要的资源 |
| 每训练num（num>=2）次就在测试集上进行1次测试；  根据所有测试中的结果保存最优模型参数 | 结合了上面两种方法的优点：  既能够及时反映模型训练的效果变化情况，又能够节省资源 | |

1. 继续完成毕设
2. 完成情况：重新阅读了论文，发现最后综合症MLP层那里忘记加了relu层，正则化邻接矩阵的方法与论文不一致，SMGCN论文结构差0.8左右，NGCF差1.0左右；代码结构进行了优化；
3. 实验结果：

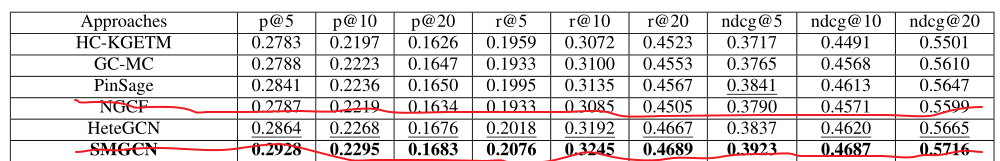
SMGCN：



NGCF：



论文结果：



1. 学习问题
2. deeplab v2+sgr的实验没有再改代码了，能运行但结果奇怪；上周已准备了PPT，且关于论文的模型实现公式已多次验证，PPT中给出了具体实现过程
3. 毕设剩HeteGNN方法没写；另一个利用知识图谱的主题模型原先没打算完成，看后面时间再决定，这个方法实现起来最难但是效果最差
4. 学习计划和安排
5. 对SMGCN和NGCF参数进行细调；完成毕设HeteGNN代码，并完善之前代码的数据展示部分，即tf.summary部分
6. 辅助学姐完成相关工作