Class MF:

**def** \_create\_placeholders:

初始化定义：user\_input(用户输入)，item\_input\_pos（有关项目），item\_input\_neg（无关项目）为[None,1]

**def** \_create\_variables：

创建：embedding\_P = (users, embedding\_size)

embedding\_Q = (items, embeddingz\_size)

delta\_P = (users, embedding\_size) #全零数组，embedding\_P的噪音

delta\_Q = (items, embedding\_size) #全零数组，embedding\_Q的噪音

h=[embedding\_size,1] #全1.0张量

**def** \_create\_inference（输入参数：item\_input\_pos）

修改：embedding\_P = (None, embedding\_size)

embedding\_Q = (None, embedding\_size)

返回：tf.matmult(P\*Q,h)和矩阵P和Q

**def** \_create\_inference\_adv

构造添加噪音的embedding\_p,embedding\_Q = (None, embedding\_size)

对抗性噪音：P\_plus\_delta Q\_plus\_delta = (None, embedding\_size)

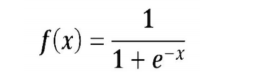
返回：tf.matmul(P\_plus\_delta \* Q\_plus\_delta, self.h)和矩阵P,Q的噪音

#构建损失函数

**def** \_create\_loss

利用：

Result = output – output\_neg shape=(None, embedding\_size)

由sigmoid函数：构建loss

加入正则化，由： 

求得opt\_loss

If self.adver:

构造加入噪音的损失函数：result\_adv

构造加入正则化的损失函数：

#创建对抗权重

**def** \_create\_adversarial:

1. 随机法生成:random

adv\_P = (users, embedding\_size)

adv\_Q = (items, embedding\_size)

L2范化法修改对抗权重为：update\_P , update\_Q

1. 基于梯度的方法生成对抗权重：grad

得到update\_P , update\_Q

**def** \_create\_optimizer:

优化器Adagrad优化参数

**def** build\_graph:

调用Class MF里的所有函数

**def** training

初始化保存操作

#训练开始

**def** training：

主函数：

初始化参数

初始化记录（日志）

初始化数据集

args.adver = 0 #0代表BPR优化模型，1表示用APR优化

初始化MF\_BPR函数

MF\_BPR.build\_graph()

开始训练：

training(MF\_BPR, dataset, args, epoch\_start=0, epoch\_end=args.adv\_epoch-1, time\_stamp=time\_stamp)