方案介绍

我们的方案是使用word2vec训练各序列的词向量拼接后使用序列模型训练模型，模型训练时使用20分类，而在预测时进行聚合，即

$$P(gender=1)=\sum_{i=1}^{10}P(gender=1,\ age=i)$$

方案中的主要部分是依靠多种手段构造差异性来进行融合（还有许多细节无法在此一一描述）：

- 模型方面：使用了多种模型，最后使用的模型包括双层双向lstm、单层transformer+单层双向lstm、单层transformer+双层双向lstm、双层transformer、dnn等。
- 序列方面：同一用户、同一个日期下的广告进行shuffle。同时，对于长度太长的序列，有的模型截左边，有的模型截右边，有的模型模型随机抽取。
- 词向量方面：以不同的参数（包括维度、窗口和负采样个数这三个参数）训练多组词向量，然后让使用不同的词向量来训练多组模型。同时部分模型会调整不同序列的词向量长度比，部分模型会使用去低频的词向量，部分模型会把一些序列的词向量设成在模型中可更新，部分模型会在训练word2vec时设置成固定窗口长度（原版word2vec的窗口长度在训练过程中是随机的）。
- 训练策略方面：我们的模型在线下一般训练3或4个epoch达到最优，但我们会多训练一个epoch使其适当过拟合。这样尽管线下会掉一个千分位左右，但有利于融合。

具体来说，我们使用的模型为（括号中为该模型的负责人）

- 双层双向lstm (jinzhen)：
  - 分别使用窗口长度为4/8/16/32/64/128的词向量(素材id/广告id 256维度，广告主id 384维，商品id 128维，商品类别和行业各64维，共896维)各训练一个模型，共6个
- 单层transformer+单层lstm (jinzhen)：
  - 分别使用窗口长度为4/8/16/32/64/128的词向量(素材id/广告id 256维度，广告主id 384维，商品id 128维，商品类别和行业各64维，共896维)各训练一个模型，共6个
  - 分别使用窗口长度为4/8/16/32/64的词向量(素材id/广告id 256维度，广告主id 384维，商品id 128维，商品类别和行业各64维，共896维)各训练一个模型，其中广告主id和商品id的向量设成可训练的，共5个模型
- DNN (jinzhen)：
  - 分别使用窗口长度为4/8/16/32/64/128的词向量(素材id/广告id 512维度，广告主id 512维，共1024维)各训练一个模型，共6个
  - 分别使用窗口长度为4/8/16/32/64/128且去低频（min-count=4）的词向量(素材id/广告id 512维度，广告主id 512维，共1024维)各训练一个模型，共6个
- 单层transformer+双层lstm (ningyu):
  - 分别使用窗口长度为6/16/32/64的词向量(creative_id 256维、advertiser_id 256维)、长度为128的点击序列分别从开始和末尾截断，训练模型，共2*4=8个
  - 分别使用窗口长度为6/16/32/64的词向量(creative_id 256维、advertiser_id 256维并且其embedding矩阵可训练)、长度为128的点击序列分别从开始和末尾截断，训练模型，共2*4=8个*
- 双层transformer (ningyu):
  - 分别使用窗口长度为6/16/32/64的词向量(creative_id 256维、advertiser_id 256维)、长度为128的点击序列分别从开始和末尾截断，训练模型，共2*4=8个

总共 6+6+5+6+6+8+8+8=53 个模型，每个模型的输出概率取平均作为最终的预测概率。

代码结构说明

- `output_prediction.py` 对于本地训练得到的概率矩阵文件输出结果
- `get_final_submission.py` 根据下载的概率矩阵文件输出结果

此外，src文件夹中包含jinzhen和ningyu两个文件夹，分别是两个团队成员的代码。时间有限，两个人的代码并未进行整合和风格统一。

代码中的各种路径（原始数据路径、临时文件路径等）是在代码文件中配置的，但不建议修改。代码运行的路径要求在复现过程中说明。

jinzhen

- `cos_uploader.py` 用于评测将结果文件上传到cos，复现用不到
- `taac_var.py` 主要是一些路径
- `dataset.py` 跑模型使用的数据集定义
- `modeling.py` 模型定义
- `preprocess.py` 对原始数据做一些预处理
- `preword2vec.py` 生成用于训练word2vec的语料
- `model_main.py` 训练和预测模型
- `word2vec/word2vec.c` 从最早的google发布的c版本的word2vec修改而来，主要加了三个东西：提供固定窗口的选项（正常word2vec的窗口是会随机变动的）；修改训练时的日志debug信息显示，增加训练时间等信息；将结果直接保存为npy文件(npy格式定义)和词表
- `word2vec/*.sh` 训练word2vec

ningyu

用于import的文件：

- `taacalgo/util.py`: 用于路径配置
- `taacalgo/dao.py`: 用于run中数据导入
- `taacalgo/w2v.py`: 用于配置word2vec以及训练过程
- `taacalgo/model.py`: 用于配置模型以及训练过程

用于运行的文件：

- `run_w2v.py`: 用于训练word2vec，生成序列与词向量矩阵。
- `run_model.py`: 用于训练模型，生成20分类的概率矩阵。
- `run_blend.py`: 用于融合模型，生成 /tmp_data/model24.npy

运行环境

- linux平台，主流发行版应该都行
- 第三方包，conda有的话就用conda装，没有的话用pip装
  - python=3.6.5
  - numpy=1.18.5
  - pandas=1.0.5
  - scikit_learn=0.23.1
  - pytorch=1.4.0
  - gensim=3.8.3

    - tqdm=4.46.1
    - cos-python-sdk-v5

复现过程

复现直接运行 `run.sh` 文件即可，此文件会下载概率矩阵文件并进行集成得到最终结果，几点说明：

- 本代码复现的是我们的次高分1.483146，仍然比后一名高，最高分的结果的融合比例和所用结果比较混乱，已经难以整理
- 为了缩小概率矩阵文件的大小，我们使用 `np.float16` 来存储概率，有一定精度损失，会造成复现得到的文件和实际复赛中提交的结果有些许差异（自己比较的话，gender的结果完全一样，age的结果只有26个label不同，对分数影响极小）
- 单模结果是随便选的，不一定是所有的模型中最好的。

如果需要从训练过程开始复现的话，创建一个和src文件夹同一级的文件夹rawdata，里面放原始数据，文件夹的结构如下：

- rawdata
    - train
        - preliminary
            - *.csv
        - semi_final
            - *.csv
    - test
        - *.csv

然后运行 `old_run.sh` 文件即可，该文件会串行调用两个人代码中的 `run.sh` ，然后根据生成的概率文件融合并生成最终结果。

代码中不同人写的代码可以并行，同一个人的模型训练过程也可以并行，但代码中都是按串行写的。

时间仓促没有进行完整调试，可能存在一些bug，如果复现过程中出现各种问题可以联系我们

- 林金镇：QQ 935410411 微信jinzhen_lin
- 宋宁宇：QQ 645205514 微信ningyu-song